Dokument Typ:
Document Type: Software User Manual

Titel:
Title:

CGS 7.3.2 User Manual

Lieferbedingungs-Nr.:
DRL/DRD No.: N/A

Klassifikations Nr.:
Classification No.: N/A

Produktgruppe:
Product Group: CGS–RIBRE–SUM–0001

Konfigurationsteil-Nr.:
Configuration Item No.: 1130992

Schlagwörter:
Headings: CGS User Manual

Produktklassifizierungs-Nr.:
Classifying Product Code:

Freigabe Ordnungs-Nr.:
Release Order No.: N/A

Bearbeitet: Prepared by:	CGS – Team	**Abteilung:** Department:	TEB33	**Firma:** Company:	EADS Astrium Space Transportation
Geprüft: Agreed by:		**Abteilung:** Department:		**Firma:** Company:	
Genehmigt: Approved by:		**Abteilung:** Department:		**Firma:** Company:	
Genehmigt: Approved by:	St. Marz	**Abteilung:** Department:	TEB33	**Firma:** Company:	EADS Astrium Space Transportation

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
2/–	29.08.2003		Updates acc. to CGS_V6.1.1
		all chapters	Editorial harmonisation of chapter headings and figure captions
		1.1	New definition of scope
		2	Update of applicable and referenced documents
		3.1	update H/W and S/W platform requirements
		3.2	update CIS
		4	Update figures
		4.2	Update interaction with CGS start/stop and selection options
		4.3	Update menus
		5.2.5	Update
		6.3	Update figure
		6.4.1	Update
		6.4.2	Update
		7.2	Update
		7.3	Update
		8	Update
		10	Adapt SUN and HP and Linux information
			Correct pathnames
			Delete SDE related text and tables
			Editorial cleaning
		Appendix A	Add new or missing abbreviations
		Appendix D	Delete FWDU related messages
			Delete Model Development related messages
			Delete Model Execution related messages
			Correct references
		Appendix I	Update
		Appendix N	Update
		Appendix O	New CIS Interface
3/–	29.04.2004		
		Appendix A	Updated
		Appendix I	Updated GROUND_LIBRARY.READ_MESSAGE_FROM_USER (SPR–14638) .
		Appendix K	Update
		Appendix N	Update

Copyright per DIN 34

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
		2.1	Update
		4.2	Update
		4.3	Update
		6.1	Update workstation types
		6.5	Update
		7.2	Update and restructure
		7.3	Update and restructure
		8	Update
		9.2.7	Update
		10	Update
3/1		7.3.2.4.5	Update
		4.2	Add overall CGS task flow and type description
		4.3	Message handling concept updates
		7.3	UCL browser and debugger updates
		Appendix I	Update
		Appendix K	Update
3/2		Appendix H	Added Load_UCL and Start_UCL_Debugger.
4/–		2.3.1	Reference to Start Center User Manual
		4.2.3	Start Center as an alternative task selection interface
		5.2.5.10	CLS Editor invocation interface updated
		Appendix I	Update
6.3.2		7.3.2.4.4	SPR–16881: renamed GoNoGo Window to Out Of Limit Display
		7.3.2.4.1	Added section for ADU Status window
6.4.0	27.06.2007	all	Removed Rev. Bars
	06.08.2007	7.3.2.4.13.4	Added section Importing USS Displays from File System into MDB
	24.08.2007	7.3.2.4.7	New color coding of monitoring display.
	10.10.2007	7.3.2.4.14.5	New section: Updating Parameter References of USS Displays in MDB
	16.10.2007	7.3.2.10	Updated icon images of AP Status and Execution Node Status
7.0.0	09.01.2008	7.3.2.6	Added section.
	18.03.2008	7.3.2.4.14.1	Integration USS/CGS: USS directory configuration.
7.2.0	23.03.2009	7.3.2.5	Virtual desktops in screen setups.
7.2.0	01.04.2009	7.3.2.4.5, 7.3.2.4.7	Apply Monitoring Window and Out Of Limit Display properties as default.

Copyright per DIN 34

DCR-Daten/Dokument-Änderungsnachweis /Data/Document Change Record

Überarbeitung Revision	**Datum** Date	**Betroffener Abschnitt/Paragraph/Seite** Affected Section/Paragraph/Page	**Änderungsgrund/Kurze Änderungsbeschreibung** Reason for Change/Brief Description of Change
7.2.0	27.04.2009	7.3.2.5	Added description how to load default screen setup. Updated screen shots.
7.2.0	27.04.2009	7.3.2.1	Removed user private services from main menu.
7.2.0	26.05.2009	7.3.2.2	Corrected description of HLCL login sequence configuration.
7.2.0	17.12.2009	7.3.2.4.1	ADU Status Display: archiving status
7.2.0	25.01.2010	7.3.2.8	Removed section User Tools
7.2.1	06.04.2010	App. N	Changed CGS–ERR–0056
7.3.0	15.04.2010	7.3.2.4.14.1	USS Integration / HLCL login sequence
7.3.0	02.02.2011	6, 7, 8, 9, B, D	Removed Final Archive SAS, Archiving and Restore.
7.3.1	10.05.2012	App. I	Changed description of ground library and HLCL CSS commands
7.3.1	14.06.2012	8.7.3	Changed description of TEV export/import tool
7.3.2	29.01.2013	7.3.2.2	Added file menu description of HLCL Command window.
	31.01.2013	7.3.2.4.11	Described search function of the Explorer window.

Copyright per DIN 34

CONTENTS

1 INTRODUCTION 1–1
1.1 Identification and Scope 1–1
1.2 Purpose 1–1
1.3 How this document is organised 1–1

2 DOCUMENTS 2–1
2.1 Applicable Documents 2–1
2.2 CGS User Manuals 2–1
2.2.1 MDA Manuals 2–1
2.2.2 WDU Manuals 2–2
2.3 Referenced User Manuals 2–3
2.4 Other Referenced Documents 2–3

3 CGS – THE DEVELOPMENT AND TEST SUPPORT SYSTEM 3–1
3.1 Overview of CGS 3–1
3.1.1 Function and Purpose 3–1
3.1.2 CGS Hardware and Commercial Software Environment 3–3
3.1.2.1 Sun Server Platform 3–3
3.1.2.2 Linux Server Platform 3–3
3.1.3 CGS Communication with a Test and Control Environment 3–3
3.2 The CGS Checkout and Test System 3–6
3.2.1 General 3–6
3.2.2 Basic Concepts, Tasks, and Features 3–6
3.2.2.1 Tailoring the Checkout and Test System to a Specific Test System Configuration 3–8
3.2.2.2 Preparing a Specific Test 3–9
3.2.2.3 Database Driven System 3–9
3.2.2.4 Distributed Configuration Concept 3–10
3.2.2.5 The Test System Open Architecture 3–13
3.2.2.6 Supported Modes for Test Operation Tasks and Activities 3–14
3.2.2.7 Test System Control by Dedicated Languages 3–16
3.2.2.7.1 Use of the User Control Language (UCL) 3–16
3.2.2.7.2 Use of the High Level Command Language (HLCL) 3–18
3.2.2.8 Test Evaluation Tools for On–Line or Off–Line Data Evaluation . . . 3–18
3.2.2.9 Test Evaluation 3–20

4 CGS GENERAL TASKS 4–1
4.1 Prerequisites 4–1
4.2 Starting CGS and CGS Task Selection 4–2
4.2.1 Message Window 4–2
4.2.2 Overall CGS Task Flow and Task Types 4–3

5 MISSION PREPARATION TASKS 5–1

Copyright per DIN 34

5.1 MDB Mission Configuration 5–2
5.1.1 Conceptual Introduction 5–2
5.1.1.1 Mission Database Structure 5–2
5.1.1.2 Version Control of Mission Configurations 5–3
5.1.2 How to Build a Mission Configuration 5–4
5.1.2.1 Starting a Mission Configuration Session 5–4
5.1.2.2 Navigation within Element Configuration Trees 5–6
5.1.2.3 Creating Nodes in a Element Configuration Tree 5–10
5.2 MDB Creation and Contents Definition of Data 5–17
5.2.1 Introduction 5–17
5.2.2 How to Define End Items Containing Data 5–18
5.2.3 How to Maintain the End Items Data (Interactive) 5–21
5.2.4 How to Maintain the End Items Data via Batch Operation 5–21
5.2.5 Defining CLS Related End Items 5–22
5.2.5.1 Introduction 5–22
5.2.5.2 Purpose of APs 5–22
5.2.5.3 How to Develop APs 5–23
5.2.5.4 Developing a User Library 5–31
5.2.5.5 Developing a System Library 5–32
5.2.5.6 Developing an HLCL Sequence 5–33
5.2.5.7 Developing a CPL Script 5–35
5.2.5.8 Developing an Item with Parameter List 5–36
5.2.5.9 Developing an Expression for Derived Values 5–36
5.2.5.10CLS Editor Invocation Interface 5–37

6 TEST AND CONTROL PREPARATION 6–1
6.1 Creating a New MDB Configuration 6–1
6.1.1 Organising the Data 6–1
6.1.2 System Prerequisites 6–2
6.1.2.1 The SYSTEM_TOPOLOGY_TABLE 6–2
6.1.2.2 Granting privileges to CGS user 6–3
6.1.3 Creating a New MDB Configuration – Defining DB End Items 6–4
6.1.3.1 Defining the Node List 6–4
6.1.3.2 Defining the Test Configuration 6–7
6.1.4 Defining House Keeping (HK) Values 6–14
6.1.5 Defining UCL Libraries 6–15
6.1.5.1 Defining UCL System Libraries for VICOS / TES 6–15
6.1.5.2 Defining UCL User Libraries 6–16
6.1.6 Deleted 6–17
6.1.7 Defining the User Specific Configuration 6–17
6.1.8 Defining SASs 6–17
6.1.9 Building a CCU 6–17
6.1.10 Performing Consistency Checks 6–18
6.1.11 Generating the Scoe Load File 6–19
6.1.12 List of Available TES HK DATA 6–21
6.2 Preparing Special Application Software 6–31

Copyright per DIN 34

6.2.1 Introduction 6–31
6.2.1.1 SAS Concept in CGS 6–31
6.2.1.1.1 TES SAS 6–31
6.2.1.1.2 TEV SAS 6–32
6.2.1.2 SAS Implementation Rules 6–33
6.2.2 The SAS Main Program 6–35
6.2.2.1 General Structure in a Non–X–View Environment 6–36
6.2.2.1.1 Synchronous SAS Main Program Structure 6–36
6.2.2.1.2 Asynchronous SAS Main Program Structure 6–38
6.2.2.2 General Structure in an X–View Environment 6–39
6.2.2.3 Handling Synchronous IO with Front End Devices 6–40
6.2.2.4 Handling Asynchronous IO with Front End Devices 6–40
6.2.3 How to Implement SAS 6–41
6.2.3.1 Implementing a TES_SAS 6–41
6.2.3.2 Implementing a TEV_SAS 6–41
6.2.4 How to Control TES SAS 6–42
6.3 Preparing Ground Synoptic Displays 6–45
6.4 Developing Simulation Models 6–46
6.4.1 MDE–GL Language Elements 6–46
6.4.1.1 Composite Function Blocks 6–46
6.4.1.2 Atomic Function Blocks 6–46
6.4.1.3 Parameter Blocks 6–46
6.4.1.4 Interface Items 6–46
6.4.1.5 Simple Connections 6–48
6.4.1.6 Logical Groupings 6–48
6.4.1.7 Global Symbols 6–48
6.4.1.8 Frame Synchronization Points 6–49
6.4.2 Model Execution Strategy 6–49
6.4.2.1 Input Activation Modes 6–49
6.4.2.2 Synchronous Function Blocks 6–50
6.4.2.3 Asynchronous Function Blocks 6–51
6.4.2.4 Connection to an external system (H/W in the loop) 6–54
6.4.2.5 The Simulation State 6–54
6.4.3 Implementation Of Atomic Functions 6–56
6.4.3.1 Description of AIL 6–56
6.4.3.2 Atomic Implementation by Decision Tables 6–65
6.4.4 Model Development Pre–requisites 6–67
6.4.4.1 Starting a Model Editing session 6–67
6.4.5 Starting CSS User Interfaces 6–70
6.4.5.1 Restrictions on model editing 6–72
6.4.6 Database Browser User Interface 6–73
6.4.6.1 General 6–73
6.4.6.2 DBB Master Window 6–73
6.4.6.3 Accessing Simulation Models 6–75
6.4.6.4 Selecting an Onboard Item 6–79
6.4.7 MDE User Interface 6–80

Copyright per DIN 34

6.4.7.1 Composite Editor 6–80
6.4.7.1.1 Basics 6–80
6.4.7.1.2 The Composite Editor’s Components 6–81
6.4.7.1.2.1 The Label 6–81
6.4.7.1.2.2 The Graphic Subview 6–81
6.4.7.1.2.3 The Tool Buttons 6–81
6.4.7.1.2.4 The Message Subview 6–82
6.4.7.1.2.5 The Overview Mode 6–82
6.4.7.1.3 Selection Sensitive Menus 6–83
6.4.7.1.4 The Composite Editor’s Basic Menu 6–84
6.4.7.1.5 Creating Block Objects, Grouping Entries, Global Symbols and Frame Synchronization Points 6–84
6.4.7.1.6 Creating Outputs, Inputs, Grouping Links 6–86
6.4.7.1.7 Selecting Objects 6–87
6.4.7.1.8 Deselecting Objects 6–87
6.4.7.1.9 Renaming Objects 6–88
6.4.7.1.10 Resizing Block Objects 6–88
6.4.7.1.11 Rotating Block Objects 6–88
6.4.7.1.12 Moving Block Objects 6–88
6.4.7.1.13 Moving I/O–Items 6–89
6.4.7.1.14 Placing Block Objects into the foreground resp. background . . 6–89
6.4.7.1.15 Removing Block Objects 6–89
6.4.7.1.16 Removing I/O–Items 6–89
6.4.7.1.17 Copying Block Objects 6–89
6.4.7.1.18 Copying Block Objects per reference 6–89
6.4.7.1.19 Changing the Activation Characteristics of Inputs 6–90
6.4.7.1.20 Stepping Through Composite Hierarchies 6–90
6.4.7.1.20.1 Stepping Into a Lower Level Composite Function Block 6–90
6.4.7.1.20.2 Stepping Back to the Next Higher Level 6–91
6.4.7.1.20.3 The Tree Browser 6–91
6.4.7.1.21 Connecting Objects 6–91
6.4.7.1.22 Connecting top level I/Os to onboard items 6–92
6.4.7.1.23 Selecting a Connection 6–94
6.4.7.1.24 Deselecting a Connection 6–95
6.4.7.1.25 Removing a Connection 6–95
6.4.7.1.26 Disconnecting an i/o–item 6–95
6.4.7.1.27 Moving Connection Lines 6–95
6.4.7.1.28 Splitting Connection Lines 6–95
6.4.7.1.29 Manipulating connected objects 6–96
6.4.7.1.30 Logical Groupings 6–96
6.4.7.1.31 Global Symbols 6–98
6.4.7.1.32 Editing an Atomic Function Block 6–100
6.4.7.1.33 Changing the Grid 6–100
6.4.7.1.34 Changing the Size of a Block’s Inside View 6–101
6.4.7.1.35 Searching for an Object 6–101
6.4.7.1.36 Changing a Function Block’s Type 6–101
6.4.7.1.37 Defining Variables 6–102
6.4.7.1.37.1 STATECODE,VECTOR,MATRIX,RECORD as I/O . . . 6–103
6.4.7.1.37.2 The AIL data types 6–106

Copyright per DIN 34

6.4.7.1.38 Performing a Rule Check 6–110
6.4.7.1.39 Saving the Model 6–110
6.4.7.1.40 Model Compilation and Simulator Kernel Configuration 6–111
6.4.7.1.40.1 Compiling atomic function blocks 6–111
6.4.7.1.40.2 Simulator Kernel Configuration 6–112
6.4.7.1.40.3 Adaptation System Configuration 6–113
6.4.7.1.41 Printing Out a Document 6–113
6.4.7.2 Model Inspectors 6–113
6.4.7.3 Composite Interface Editor 6–115
6.4.7.4 Atomic Editors 6–117
6.4.7.4.1 Common 6–117
6.4.7.4.2 Components 6–117
6.4.7.4.2.1 The Label 6–118
6.4.7.4.2.2 The Graphic Subview 6–118
6.4.7.4.2.3 The Input and Output Lists 6–118
6.4.7.4.2.4 The Message Subview 6–118
6.4.7.4.2.5 The Implementation Subview 6–118
6.4.7.4.3 Atomic AIL Editor 6–119
6.4.7.4.3.1 AIL special features (hibernate, PULSE type parameter) 6–120
6.4.7.4.3.2 Variables in the AIL code 6–122
6.4.7.4.3.3 Restrictions 6–123
6.4.7.4.4 Atomic Decision Table Editor 6–124
6.4.7.5 Icon Editor 6–130
6.4.7.5.1 Basics 6–130
6.4.7.5.2 Components 6–131
6.4.7.6 Tree Browser 6–133
6.4.7.7 Documentation Generation Function 6–135
6.4.7.8 Onboard References Adaptation Function 6–138
6.4.7.8.1 Basics 6–138
6.4.7.9 The Simulation Table Editor Window 6–140
6.4.7.9.1 Creating a Simulation Table 6–140
6.4.7.9.2 Creating monitoring items in the simulation table 6–142
6.4.7.9.3 Restrictions on monitoring 6–146
6.4.7.9.4 Creating logging items in the simulation table 6–147
6.4.7.9.5 Restrictions on logging 6–150
6.4.7.9.6 Creating tracing items in the simulation table 6–150
6.4.7.9.7 Restrictions on tracing 6–152
6.4.8 CSS Configuration Environment Variables 6–153
6.4.9 File System Maintenance 6–154
6.4.10 Creating Onboard End Items in the MDB 6–156
6.4.10.1 End Item Type 6–156
6.4.10.2 Mapping to CSS Data Type 6–156
6.4.10.3 Physical Address 6–159
6.4.10.4 CCSDS TM/TC Packets 6–161
6.4.10.5 Calibration/Decalibaration Definition 6–162
6.4.10.6 Feedback 6–163
6.4.11 Description of mathematical constants and routines delivered with CSS 6–164
6.4.12 CSS Data Types 6–174

Copyright per DIN 34

6.5 Model Observation & Control 6–185
6.5.1 Basics 6–185
6.5.2 Starting MOCS from I_MDB 6–186
6.5.3 Starting the simulator 6–189
6.5.4 User authorisation 6–191
6.5.5 Starting a simulation session 6–194
6.5.6 Simulation execution 6–198
6.5.6.1 The Session Observer window 6–198
6.5.6.2 Basics 6–198
6.5.6.3 On–line monitoring 6–199
6.5.6.4 Time scales used during simulation execution 6–203
6.5.6.4.1 Setting the Simulated Mission Time 6–203
6.5.6.4.1.1 Setting the SMT starting point 6–203
6.5.6.4.1.2 Setting the SMT increment per minframe 6–204
6.5.6.5 The non–Real Time Simulation Modes 6–204
6.5.6.5.1 The Minframe Mode 6–205
6.5.6.5.2 The Processing Mode 6–205
6.5.6.6 The Real Time Simulation Mode 6–206
6.5.6.7 Creating new Simulation States 6–206
6.5.6.8 Errors during simulation execution 6–207
6.5.7 Commanding CSS via HLCL primary commands 6–207
6.5.8 Simulation results evaluation 6–209
6.6 CGS and Application Domain Specific End Items 6–211
6.6.1 General 6–211
6.6.2 End Item Concept and Usage 6–211
6.6.3 Semi–Formal Description of End Items, Aggregates and Attributes 6–211
6.6.3.1 End Item Editing via Interactive Tools 6–212
6.6.3.2 End Item Creation via Batch Processing 6–212
6.6.3.3 Special Tools for End Item Editing 6–212
6.6.4 Basic End Item Type Definitions 6–212
6.6.4.1 Basic MDB Types and Enumerations 6–212
6.6.4.2 MDB Domain and Application Domain End Item Association 6–216
6.6.4.2.1 Domain Specific End Items in CGS 6–216
6.6.4.2.1.1 End Items in the Domain CGS 6–216
6.6.4.2.1.2 End Items in the Domain CSS 6–219
6.6.4.2.1.3 End Items in the Domain CSS_ONBOARD 6–219
6.6.4.2.1.4 End Items in the Domain EGSE 6–220
6.6.4.2.1.5 End Items in the Domain SDDF 6–222
6.6.4.2.1.6 End Items in the Domain UCL_LIBRARY 6–223
6.6.4.2.1.7 End Items in the Domain DEMO_ON_BORD 6–223
6.6.5 End Item Type Descriptions 6–225
6.6.5.1 Aggregates Common to all End Items 6–225
6.6.5.2 Description of End Item Groups 6–225
6.6.5.2.1 Measurements 6–226
6.6.5.2.1.1 EGSE_integer_measurement 6–227
6.6.5.2.1.2 UNSIGNED_integer_measurement 6–227
6.6.5.2.1.3 EGSE_FLOAT_MEASUREMENT 6–227

Copyright per DIN 34

6.6.5.2.1.4 DOUBLE_FLOAT_MEASUREMENT 6–227
6.6.5.2.1.5 EGSE_discrete_measurement 6–227
6.6.5.2.1.6 EGSE_Bytestream_measurement 6–228
6.6.5.2.1.7 BOOLEAN_MEASUREMENT 6–228
6.6.5.2.2 SW Variables 6–229
6.6.5.2.2.1 EGSE_INTEGER_SW_VARIABLE 6–230
6.6.5.2.2.2 UNSIGNED_INTEGER_SW_VARIABLE 6–230
6.6.5.2.2.3 EGSE_FLOAT_SW_VARIABLE 6–230
6.6.5.2.2.4 DOUBLE_FLOAT_SW_VARIABLE 6–230
6.6.5.2.2.5 EGSE_DISCRETE_SW_VARIABLE 6–230
6.6.5.2.2.6 EGSE_BYTESTREAM_SW_VARIABLE 6–230
6.6.5.2.2.7 BOOLEAN_SW_VARIABLE 6–231
6.6.5.2.3 Derived Values 6–231
6.6.5.2.3.1 EGSE_INTEGER_DERIVED_VALUE 6–233
6.6.5.2.3.2 UNSIGNED_INT_DERIVED_VALUE 6–233
6.6.5.2.3.3 EGSE_FLOAT_DERIVED_VALUE 6–233
6.6.5.2.3.4 DOUBLE_FLOAT_DERIVED_VALUE 6–233
6.6.5.2.3.5 EGSE_DISCRETE_DERIVED_VALUE 6–233
6.6.5.2.3.6 EGSE_STRING_DERIVED_VALUE 6–233
6.6.5.2.4 Messages 6–234
6.6.5.2.4.1 EGSE_USER_MESSAGE 6–234
6.6.5.2.5 Test Facility Description 6–234
6.6.5.2.5.1 EGSE_NODE 6–234
6.6.5.2.5.2 EGSE_SOFTWARE 6–234
6.6.5.2.5.3 EGSE_TEST_CONFIGURATION 6–234
6.6.5.2.6 Generation Data Units, Stimuli and Telecommands 6–235
6.6.5.2.6.1 EGSE_ANALOG_STIMULUS 6–237
6.6.5.2.6.2 EGSE_DISCRETE_STIMULUS 6–237
6.6.5.2.6.3 EGSE_BINARY_PACKET 6–237
6.6.5.2.6.4 EGSE_PREDEFINED_TC 6–239
6.6.5.2.6.5 PUS_TC 6–241
6.6.5.2.6.6 INTEGER_STIMULUS 6–242
6.6.5.2.6.7 UNSIGNED_INTEGER_STIMULUS 6–242
6.6.5.2.6.8 DOUBLE_FLOAT_STIMULUS 6–242
6.6.5.2.6.9 BOOLEAN_STIMULUS 6–243
6.6.5.2.6.10 PULSE_STIMULUS 6–243
6.6.5.2.6.11 BURST_PULSE_STIMULUS 6–243
6.6.5.2.7 SWOP Commands and Response Packets 6–243
6.6.5.2.7.1 SWOP_COMMAND 6–243
6.6.5.2.7.2 RESPONSE_PACKET 6–243
6.6.5.2.7.3 APPLICATION_ID (APID) 6–244
6.6.5.2.7.4 CCSDS_END_POINT 6–244
6.6.5.2.8 Acquisition Data Units 6–244
6.6.5.2.8.1 STRUCTURED_ADU_DESCRIPTION 6–245
6.6.5.2.8.2 UNSTRUCTURED_ADU_DESCRIPTION 6–245
6.6.5.2.8.3 CCSDS_ADU_DESCRIPTION 6–245
6.6.5.2.8.4 PUS_ADU_DESCRIPTION 6–245
6.6.5.2.9 Simulated Data 6–246
6.6.5.2.9.1 SIMULATED_ADU_DESCRIPTION 6–246

Copyright per DIN 34

6.6.5.2.10 Lists 6–246
6.6.5.2.10.1 EGSE_MONITOR_LIST 6–246
6.6.5.2.10.2 PUS STRUCTURE_ID 6–246
6.6.5.2.10.3 GDU_DESCRIPTION_LIST 6–246
6.7 Consistency Checking 6–247
6.7.1 Input Checking 6–247
6.7.2 Item Checking 6–247
6.7.2.1 Consistency Checker on End Item Level 6–247
6.7.2.2 Check_MDB_Item Program 6–247
6.7.3 Consistency Checking on CDU/CCU Level 6–248
6.7.4 Checking when Loading to Files 6–248
6.8 CIS Interface 6–249
6.8.1 CIS XML Schema Definitions 6–250

7 INTEGRATION AND TEST 7–1
7.1 General Operation in the Checkout Environment 7–1
7.1.1 Checkout Operations 7–1
7.1.2 Operational Modes 7–1
7.1.3 Operational Configurations 7–2
7.1.4 Operational Constraints 7–4
7.2 Setting-Up the Test Environment 7–6
7.2.1 Introduction 7–6
7.2.2 Test System Configuration and Verification (TSCV) 7–6
7.2.2.1 Selected CCU Version 7–6
7.2.2.2 Describing and Configuring a Test Configuration 7–7
7.2.2.3 Setup a Test Configuration 7–7
7.2.2.4 Test Node States and Control 7–8
7.2.2.5 Test Session States and Control 7–9
7.2.2.5.1 Deleting Test Sessions 7–10
7.2.2.6 System Table Maintenance 7–11
7.2.2.7 System Services Management 7–11
7.2.2.8 Operations Environment 7–11
7.2.2.8.1 TSCV Operations Constraints 7–11
7.2.2.9 Operation Basics 7–11
7.2.2.9.1 Test System Control and Configuration 7–12
7.2.2.9.2 Test Configuration Management 7–12
7.2.2.9.3 Test Configuration Control 7–13
7.2.2.9.4 Test Configuration Edit 7–14
7.2.2.9.5 Test Session Control 7–15
7.2.2.9.6 Replay Session Control 7–15
7.2.2.9.7 Test Configuration Application Control 7–17
7.2.2.10 TSCV Operation Control Procedures And Instructions 7–18
7.2.2.10.1 Interactive Mode 7–18
7.2.2.10.2 Batch Mode 7–18
7.2.2.10.3 TSCV Housekeeping 7–18
7.2.2.11 TSCV Reference Information 7–19
7.2.2.11.1 Help Method 7–19

Copyright per DIN 34

7.2.2.11.2 Screen Definitions and Operations 7–19
7.2.3 The TSCV Main Window 7–20
7.2.3.1 General 7–20
7.2.3.2 The ’System’ Menu 7–22
7.2.3.3 The ’System–>Launch Services’ Menu Option 7–22
7.2.3.4 The ’System–>Shutdown’ Menu Option 7–22
7.2.3.5 The ’Properties’ Menu 7–23
7.2.3.6 The ’Properties–>System Topology...’ Menu Option 7–23
7.2.3.7 The ’Properties–>Software Versions...’ Menu Option 7–23
7.2.3.8 The ’Configuration’ Menu 7–23
7.2.3.9 The ’Configuration–>Check Status’ Menu Option 7–23
7.2.3.10 The ’Configuration–>Setup’ Menu Option 7–24
7.2.3.11 The ’Configuration–>Start’ Menu Option 7–25
7.2.3.12 The ’Configuration–>Suspend’ Menu Option 7–25
7.2.3.13 The ’Configuration–>Resume’ Menu Option 7–25
7.2.3.14 The ’Configuration–>Stop’ Menu Option 7–25
7.2.3.15 The ’Configuration–>Shutdown’ Menu Option 7–25
7.2.3.16 The ’Edit’ Menu 7–26
7.2.3.17 The ’Edit–>Load for online test...’ Menu Option 7–26
7.2.3.18 The ’Edit–>Load for Replay...’ Menu Option 7–26
7.2.3.19 The ’Edit–>Unload’ Menu Option 7–26
7.2.3.19.1 The ’Edit–>View...’ Menu Option 7–26
7.2.3.20 The ’Test Session’ Menu 7–27
7.2.3.21 The ’Test Session–>Create’ Menu Option 7–27
7.2.3.22 The ’Test Session–>Close’ Menu Option 7–27
7.2.3.23 The ’Test Session–>Maintain’ Menu Option 7–27
7.2.3.24 The ’Replay Session’ Menu 7–27
7.2.3.25 The ’Replay Session–>Assign...’ Menu Option 7–27
7.2.3.26 The ’Replay Session–>Properties...’ Menu Option 7–28
7.2.3.27 The Test Configuration List 7–28
7.2.3.28 The ’Test Configuration’ Text Field 7–28
7.2.3.29 The ’System Tree Version’ Text Field 7–29
7.2.3.30 The ’Mission Name’ Text Field 7–29
7.2.3.31 The ’CCU Configuration’ Text Field 7–29
7.2.3.32 The ’CCU Pathname’ Text Field 7–29
7.2.3.32.1 The ’CCU Version’ Text Field 7–29
7.2.3.33 The ’Node’ Menu 7–29
7.2.3.34 The ’Node–>Check Status’ Menu Option 7–29
7.2.3.35 The ’Node–>Launch Services’ Menu Option 7–29
7.2.3.36 The ’Node–>Launch TES’ Menu Option 7–29
7.2.3.37 The ’Node–>Setup’ Menu Option 7–30
7.2.3.38 The ’Node–>Start’ Menu Option 7–30
7.2.3.39 The ’Node–>Suspend’ Menu Option 7–30
7.2.3.40 The ’Node–>Resume’ Menu Option 7–30
7.2.3.41 The ’Node–>Stop’ Menu Option 7–30
7.2.3.42 The ’Node–>Shutdown’ Menu Option 7–30

Copyright per DIN 34

7.2.3.43 The 'Properties' Menu 7–31
7.2.3.44 The 'Properties–>Node...' Menu Option 7–31
7.2.3.45 The List of Nodes 7–31
7.2.4 The Load Test Configuration Window 7–32
7.2.4.1 General 7–32
7.2.4.2 The 'Select CCU...' Button 7–34
7.2.4.3 The 'System Tree Version' Text Field 7–34
7.2.4.4 The 'CCU Configuration' Text Field 7–34
7.2.4.5 The 'CCU Pathname' Text Field 7–34
7.2.4.6 The 'CCU Version' Text Field 7–34
7.2.4.7 The 'Test Configurations' List 7–34
7.2.4.8 The 'Load' Button 7–34
7.2.4.9 The 'View...' Button 7–35
7.2.5 The Select CCU Window 7–35
7.2.5.1 General 7–35
7.2.5.2 The 'Element Configurations' Button 7–37
7.2.5.3 The 'Mission Names' Button 7–37
7.2.5.4 The 'System Tree Versions' Button 7–37
7.2.5.5 The 'CCU' Button 7–37
7.2.5.6 The 'CCU Versions' Button 7–37
7.2.5.7 The Selection List 7–38
7.2.5.8 The 'Element Configuration' Text Item 7–38
7.2.5.9 The 'Mission Name' Text Item 7–38
7.2.5.10 The 'System Tree Version' Text Item 7–38
7.2.5.11 The 'CCU Configuration Name' Text Item 7–38
7.2.5.12 The 'CCU Pathname' Text Item 7–38
7.2.5.13 The 'CCU Version' Text Item 7–38
7.2.5.14 The 'Apply' Button 7–38
7.2.5.15 The 'Reset' Button 7–39
7.2.6 The Create Session Window 7–39
7.2.6.1 General 7–39
7.2.6.2 The 'Prefix' and 'Purpose' Text Item 7–39
7.2.6.3 The 'Prefix' Text Item 7–40
7.2.6.4 The 'Purpose' Text Item 7–40
7.2.7 The 'Store on the Final Archive Medium' Check Box 7–40
7.2.7.1 The 'Disable Logging Of Data' Check Box 7–40
7.2.7.2 The 'Cyclic Session' Check Box 7–40
7.2.7.3 The 'First Switch Time' Check Box 7–40
7.2.7.4 The 'Apply' Button 7–40
7.2.7.5 The 'Reset' Button 7–40
7.2.8 The Multi Purpose Test Sessions Window 7–41
7.2.8.1 General 7–41
7.2.8.2 The 'Session Pattern' Text Item 7–43
7.2.8.3 The 'Session Mode' Choice 7–43
7.2.8.4 The 'Session Status' Choice 7–44
7.2.8.5 The 'Specific' Choice 7–44

Copyright per DIN 34

7.2.8.6 The ’Select...’ Button 7–44
7.2.8.7 The ’System Tree Version’ Text Field 7–45
7.2.8.8 The ’CCU Configuration’ Text Field 7–45
7.2.8.9 The ’CCU Pathname’ Text Field 7–45
7.2.8.10 The ’CCU Version’ Text Field 7–45
7.2.8.11 The ’Created After’ Timestamp 7–45
7.2.8.12 The ’Created Before’ Timestamp 7–45
7.2.8.13 The ’Sorting’ Choice Options 7–45
7.2.8.14 The ’List Session’ Button 7–45
7.2.8.15 The ‘Test Sessions’ List Panel 7–46
7.2.8.16 The ’View...’ Button 7–46
7.2.8.17 The ’Session Size ...’ Button 7–46
7.2.8.18 The ’Close’ Button 7–46
7.2.8.19 The ’Delete ...’ Button 7–46
7.2.8.20 The ’Delete in Default Session...’ Button 7–46
7.2.9 The Node Property Sheet Window 7–47
7.2.9.1 General 7–47
7.2.9.2 The ’Instance’ Text Field 7–48
7.2.9.2.1 The ’Path Name’ Text Field 7–48
7.2.9.3 The ’Participating’ Check Box 7–49
7.2.9.4 The ’MTP node’ Check Box 7–49
7.2.9.5 The ’Execution Mode’ Check Box – Normal or Simulation 7–49
7.2.9.6 The ’Execution Mode’ Check Box – Replay TCs 7–49
7.2.9.7 The ’Forced Loading’ Check Box 7–49
7.2.9.8 The ’Apply’ Button 7–49
7.2.9.9 The ’Reset’ Button 7–50
7.2.10 The Replay Properties Window 7–50
7.2.10.1 The ’Replay Session’ Text Field 7–51
7.2.10.2 The ’Recorded Time Frame’ Text Field 7–51
7.2.10.3 The ’Replay Speed Factor’ Text Field 7–51
7.2.10.4 The ’Time Base’ Choice 7–51
7.2.10.5 The ’User Events’ List 7–52
7.2.10.6 The ’Set Begin’ Button 7–52
7.2.10.7 The ’Set End’ Button 7–52
7.2.10.8 The ’Begin’ Timestamp 7–52
7.2.10.9 The Begin ’Event’ Text Field 7–52
7.2.10.10 The ’End’ Timestamp 7–53
7.2.10.11 The End ’Event’ Text Field 7–53
7.2.10.12 The ’Apply’ Button 7–53
7.2.10.13 The ’Reset’ Button 7–53
7.2.11 The Maintain System Topology Window 7–53
7.2.11.1 General 7–53
7.2.11.2 The ’Test Site’ Text Field 7–54

Copyright per DIN 34

7.2.11.3 The System Topology Table List 7–54
7.2.11.4 The ’Add Before’ Button 7–54
7.2.11.5 The ’Add After’ Button 7–55
7.2.11.6 The ’Delete’ Button 7–55
7.2.11.7 The ’Change’ Button 7–55
7.2.11.8 The ’Host’ Text Field 7–55
7.2.11.9 The ’Instance’ Text Field 7–55
7.2.11.10 The ’Port Number’ Text Field 7–56
7.2.11.11 The ’Apply’ Button 7–56
7.2.11.12 The ’Reset’ Button 7–56
7.2.12 The Display Request Window 7–57
7.2.13 Operator Commands and Operations 7–58
7.2.14 TSCV Invocation, Interactive Mode 7–58
7.2.15 TSCV Invocation, Batch Mode 7–59
7.2.16 The Interactive Mode 7–60
7.2.17 Batch Mode Exit Codes 7–62
7.3 Test Execution 7–63
7.3.1 Overview 7–63
7.3.1.1 Visualisation of Data 7–63
7.3.1.2 Commanding an Execution Session 7–65
7.3.1.3 System Housekeeping Data 7–66
7.3.1.4 Storing of On–line Data 7–66
7.3.1.5 Access to Stored On–line Data 7–66
7.3.1.6 Automatic Data Supervision Features 7–67
7.3.1.7 On–line Modifications of the Execution Configuration 7–67
7.3.2 Online Control Applications 7–68
7.3.2.1 Invoking Online Control (HCI) 7–69
7.3.2.2 HLCL Commanding 7–71
7.3.2.3 CGS Tools 7–77
7.3.2.4 Data Displays for Online Control 7–78
7.3.2.4.1 ADU Status 7–78
7.3.2.4.2 AP Status 7–78
7.3.2.4.3 Clock 7–79
7.3.2.4.4 Database Node Status 7–81
7.3.2.4.5 Out of Limit Display 7–82
7.3.2.4.6 Graph Facility 7–85
7.3.2.4.7 Monitoring Window 7–86
7.3.2.4.8 Raw Data Dump 7–93
7.3.2.4.9 SAS Status 7–95
7.3.2.4.10 System Advisory 7–95
7.3.2.4.11 Explorer 7–99
7.3.2.4.12 Execution Node Status 7–100
7.3.2.4.13 USS Displays 7–108
7.3.2.4.13.1 Integration with CGS 7–108

Copyright per DIN 34

7.3.2.4.13.2 GWDU to USS Display Conversion 7–109
7.3.2.4.13.3 Starting USS Editor 7–114
7.3.2.4.13.4 Importing USS Displays from File System into MDB . . 7–116
7.3.2.4.13.5 Updating Parameter References of USS Displays in MDB 7–119
7.3.2.4.13.6 Starting USS Executor 7–120
7.3.2.5 Screen Setup Maintenance 7–120
7.3.2.6 Starting Applications via HLCL/UCL 7–123
7.3.2.7 UCL Browser 7–124
7.3.2.8 Quit 7–126
7.3.2.9 Input Dialog 7–126
7.3.2.10 Telemetry Window Footer Configuration 7–128
7.3.3 Test Execution: Monitoring, Archiving and AP Execution 7–129
7.3.3.1 General 7–129
7.3.3.2 Monitoring and Data Processing 7–129
7.3.3.3 Sending of Generation_Data_Units (GDU) 7–136
7.3.3.3.1 Prevent Command Sending 7–138
7.3.3.3.2 Critical Commands 7–138
7.3.3.3.3 GDU Preconditions 7–139
7.3.3.3.4 Telecommand Verification 7–139
7.3.3.4 Archiving and Logging 7–140
7.3.3.5 UCL Execution 7–140
7.3.3.6 Communication with SAS 7–145
7.3.3.7 Replaying Data 7–145
7.3.3.8 Simulating Data 7–146
7.3.3.9 Internode Communication 7–148
7.3.3.10 The configuration of TES 7–149

8 TEST EVALUATION 8–1
8.1 General 8–1
8.2 Changes in CGS 6.3.2 8–1
8.3 Changes in CGS 7.1.0 8–2
8.4 Changes in CGS 7.3.0 8–2
8.5 Changes in CGS 7.3.1 8–2
8.6 Using TEV in Batch Mode 8–3
8.6.1 Inputs 8–3
8.6.2 Session File 8–4
8.6.3 Definition File 8–4
8.6.4 Result File 8–5
8.6.5 Error Handling 8–5
8.7 Test Evaluation Preparation 8–6
8.7.1 Evaluation Session Definition 8–7
8.7.2 Execution Session Initialization 8–11
8.7.2.1 Select execution sessions : 8–12
8.7.2.2 The Different Functionalities to Select a CCU 8–15
8.7.2.3 Validate the Selections : 8–18

Copyright per DIN 34

8.7.3 The Export/Import Tool 8–19
8.8 Data Evaluation Tools 8–21
8.8.1 Evaluation Tools Overview 8–21
8.8.2 The Definition Menu 8–21
8.8.3 The Result Menu 8–23
8.8.4 Select a Time Frame 8–24
8.8.5 Events Logging Tool 8–27
8.8.5.1 Create a Definition 8–27
8.8.5.2 Generate a Result 8–30
8.8.5.3 Generate an ADT Result 8–31
8.8.6 Raw Data Dump Tool 8–33
8.8.6.1 Create a Definition 8–35
8.8.6.2 Consult the Packets Dumped in the Archive 8–39
8.8.6.3 Consult the Packets in the Archive in Summary Format 8–41
8.8.6.4 Save the Selected Packets in a Result File 8–43
8.8.7 Data Set Generation Tool 8–44
8.8.7.1 Select the Sampling Mode 8–46
8.8.7.2 Select Data Set Parameters 8–46
8.8.7.3 Configure Time Tags 8–48
8.8.7.4 Generate a Data Set 8–50
8.8.7.5 Generate a Pre–View Listing 8–51
8.8.8 Statistics Generation Tool (in CGS 6.3.2 replaced by Data Viewer Tool) . . . 8–52
8.8.9 Data Listing Tool 8–55
8.8.10 The Result Menu 8–56
8.8.11 Graphic Tool (in CGS 6.3.2 replaced by Data Viewer Tool) 8–60
8.9 Utilities 8–68
8.9.1 File Handling 8–68
8.9.2 Data Set Merger 8–71
8.9.3 Data Set / Events Merger Tool 8–71

9 TRDB TOOLS 9–75
9.1 General 9–75
9.2 The Recovery Scripts 9–75
9.2.1 Getting Started 9–75
9.2.1.1 Execution Session 9–75
9.2.1.1.1 Execution Session Diagnostic 9–76
9.2.1.1.2 Delete Execution Session Menu 9–79
9.2.1.1.3 Session Is Used Menu 9–82
9.2.1.1.4 File Storage Failure Menu 9–87
9.2.1.1.5 Close Session 9–90
9.2.1.1.6 List Execution Sessions 9–92
9.2.1.1.7 File Transfer From Local Dir Failures 9–93
9.2.1.1.8 Close Aborted Test Session to Normally Closed 9–98
9.2.1.2 Evaluation Session Menu 9–99
9.2.1.2.1 Evaluation Session Diagnostic 9–99
9.2.1.2.2 Delete Evaluation Session Menu 9–101

Copyright per DIN 34

9.2.1.2.3 Session Is Used Menu 9–103
9.2.1.2.4 File Storage Failure Menu 9–109
9.2.1.2.5 Remove Evaluation Users Menu 9–112
9.2.1.2.6 List Evaluation Sessions 9–117
9.2.1.3 DBS Error Number (DBS_ERR_xxx) 9–118

10 CGS ADMINISTRATION 10–1
10.1 CGS System Administration 10–1
10.1.1 CGS Component Version Status Info 10–1
10.1.2 CGS File System Structure 10–1
10.1.3 CGS Configuration Support Tool 10–4
10.1.4 CGS Hardware Installation/Deinstallation And Booting 10–5
10.2 CGS User Administration 10–5
10.2.1 Add CGS User 10–5
10.2.2 Deinstall CGS User 10–5
10.2.3 CGS User Authorization 10–5
10.2.3.1 CGS System Privileges 10–5
10.2.3.2 CGS User Privileges 10–7
10.2.3.2.1 Syntax of Privilege Descriptions 10–7
10.2.3.3 The PRIV Tool for Privilege Assignments 10–7
10.3 Configuration Setup 10–10
10.3.1 Modify System Topology Table 10–10
10.3.2 Maintain CGS Configuration Parameters 10–10
10.4 Oracle Startup/Shutdown 10–11
10.5 MDB Administration 10–11
10.6 Maintain Storage Resources 10–11
10.6.1 Resource Considerations 10–11
10.6.2 Hard Disc (Magnetic Disc) 10–12
10.6.2.1 Monitoring of Disc Space 10–12
10.6.2.2 Delete/Export Test Sessions 10–13
10.6.2.3 CleanUp of Disk Memory 10–13
10.6.3 Configuration of Sizes and Estimation of TRDB Data 10–15
10.6.3.1 Size Configuration for Event Logging 10–15
10.6.3.2 Size Configuration for Engineering Value Logbooks 10–16
10.6.3.3 Configuration and Size Confuiguration of Archive Files 10–16
10.6.3.4 Result File Size and Handling 10–17
10.7 Monitor System Behaviour 10–17
10.7.1 Monitor Process Status 10–17
10.7.2 Monitor Memory Status 10–18
10.7.3 Monitor Time Synchronisation Status 10–19

A ACRONYMS A-1

B DEFINITIONS B-1

C END ITEM TYPES C-1

Copyright per DIN 34

D CGS ERROR MESSAGES D–1
D-1 Commercial Tools D–1
D-1.1 The ORACLE Database D–1
D-1.1.1 ORACLE On–Line Help Facility D–1
D-1.2 The Ada Compiler D–2
D-2 Test Preparation D–3
D-2.1 MDA Error Messages D–3
D-2.1.1 Consistency Checker Error Messages D–3
D-2.1.2 Export/import Error Messages D–3
D-2.1.3 Batch data entry Error Messages D–3
D-2.1.4 I_MDB Error Messages D–3
D-2.1.5 Generate SCOE Files D–3
D-2.1.6 GWDU: Ground Synoptic Display Editor D–4
D-2.1.7 CLS Editor and Compiler D–4
D-2.2 HLCL Command Sequences D–5
D-2.2.1 Model Development D–7
D-2.2.1.1 General D–7
D-2.2.1.2 Error messages during model editing D–7
D-2.2.1.3 Error messages during model configuration D–8
D-3 Test Setup and Execution D–11
D-3.1 VICOS Error Messages D–11
D-3.2 Test Setup D–14
D-3.2.1 System Setup / Shutdown D–14
D-3.2.2 Test Configuration Setup and Verification (TSCV) D–14
D-3.2.2.1 TCSV Error Conditions D–14
D-3.2.2.2 TCSV Error Messages in the Console Window D–16
D-3.2.2.3 TCSV Error Messages to the Message Window D–16
D-3.2.2.4 TCSV Error Messages in Pop Up Window D–22
D-3.3 Test Execution D–23
D-3.3.1 Messages from HCI (Workstation) D–23
D-3.3.1.1 Messages on Console Window D–23
D-3.3.1.2 Messages in Message Window D–24
D-3.3.2 Messages from TES (Test Node) D–26
D-3.3.3 Messages from DBS (Test Result DB) D–70
D-3.4 Model Execution D–97
D-3.4.1 HLCL on–line D–97
D-3.4.2 Model Execution Messages D–97
D-3.4.2.1 Error messages produced by DB Browser, MOCS and ICP . D–99
D-3.4.2.2 DB Browser Messages D–100
D-3.4.2.3 MDE Error Messages – created during model editing D–103
D-3.4.2.4 Compilation errors from the CSS runtime system (CTG) . . D–118
D-3.4.2.5 MOCS error messages D–120
D-3.4.2.6 Runtime Error Messages created during model execution . D–132
D-4 Test Evaluation D–138
D-4.1 TEV Error Report Basics D–138
D-4.1.1 DBS error messages displayed in TEV D–138

Copyright per DIN 34

D-4.1.2 TEV Footer Messages D–143
D-4.1.3 TEV Status Messages D–147
D-4.1.4 TEV Error Messages D–152

E NOTATIONAL CONVENTIONS E–1

F DESCRIPTION OF THE CGS TES–SAS API INTERFACE F–1
F-1 Connecting to and Disconnecting from CGS F–29
F-2 Reading and Handling Commands F–30
F-3 Handling GDUs F–38
F-4 Handling ADUs F–43
F-5 Reading End Item Data from CGS F–50
F-6 Providing End Item Data to CGS F–53
F-7 Starting Parameterless UCL Automated Procedure F–54
F-8 Exchanging Messages with APs F–54
F-9 Reading the CGS Time F–55
F-10 Reporting Errors and Passing Messages to CGS F–56
F-11 Store / Retrieve Files to Test Resul Data Base (TRDB) F–57
F-12 Get GDU Description from CGS F–58
F-13 Translation SID – Pathname/Nickname F–58

G INDEX OF ALL USER MANUAL PROCEDURES G–1

H TEST AND CONTROL RELATED HLCL COMMANDS H–1
H-1 Basics H–1
H-2 HLCL Language Definition and Primary Commands H–1
H-3 Application Specific Primary Commands H–2
H-3.1 EGSE Specific Primary Commands H–2
H-3.2 CSS Related Specific Primary Commands H–6
H-4 UCL System Libraries H–24
H-4.1 System Libraries H–24

I UCL SYSTEM LIBRARIES I–1
I-1 UCL System library: GROUND_COMMON I–3
I-1.1 UCL System library Specification GROUND_COMMON I–3
I-2 UCL System library: GROUND_library I–11
I-2.1 UCL Ground System library Specification I–11
I-3 UCL System library: GRD_CONVERSION I–33
I-3.1 UCL Ground System library Specification: GRD_CONVERSION I–33
I-4 UCL Ground Commands To Onboard System library I–34
I-4.1 UCL GROUND_TO_OB_LIB System library Specification I–34
I-5 UCL Ground System library: GROUND_VALUES I–40
I-5.1 UCL GROUND_VALUES System library Specification I–40

Copyright per DIN 34

I-6 UCL Ground System library: MATH_LIB I–63
I-6.1 UCL System library Specification I–63
I-7 UCL Ground System library: MATH_LIB_LONG I–66
I-7.1 UCL System library Specification I–66
I-8 UCL Ground System library: MESSAGE_LIBRARY I–69
I-8.1 UCL System library Specification I–69
I-9 UCL Ground System library: TC_CONSTRUCTION I–71
I-9.1 UCL System library Specification I–71
I-10 UCL Ground System library: RAW_DATA_library I–80
I-10.1 UCL System library Specification I–80
I-11 UCL Ground System library: PACKET_library I–83
I-11.1 UCL System library Specification I–83
I-12 UCL Ground System library Specification: ATV_library I–91
I-12.1 UCL System library Specification I–91
I-13 UCL Ground System library Specification: TEXT_FILE_IO I–93
I-13.1 UCL System library Specification I–93
I-14 UCL Ground System library Specification: BINARY_FILE_IO I–98
I-14.1 UCL System library Specification I–98
I-15 UCL Ground System library Specification: SYSTEM I–101
I-15.1 UCL System library Specification I–101
I-16 UCL Ground System library Specification: CALIBRATION_LIB I–103
I-16.1 UCL System library Specification I–103
I-17 HLCL Ground System library Specification: HLCL_CSS_CMDS I–112
I-17.1 HLCL System library Specification I–112
I-18 HLCL Ground System library Specification: HLCL_HCI_CMDS I–122
I-18.1 HLCL System library Specification I–122

J CGS SYSTEM LIMITATIONS J–1
J-1 Diverse CGS Constraints J–1
J-2 Miscellaneous Resources J–2

K USER DEFINABLE CONFIGURATION FILES FOR CGS K–1
K-1 Scope of Configuration Files K–1
K-2 Configuration Editor K–1
K-3 Config tool K–2
K-4 List of Configuration and Property Files K–3
K-5 Example Configuration and Property Files K–5

L CGS SCREEN SETUP L–1
L-1 Screen Setups and Window Definitions L–1

M UCL FILE IO VIA A SPECIFIC SAS M–1
M-1 Communication between FILE_IO_LIB and SAS_FILE_IO M–2

Copyright per DIN 34

M-1.1 Messages M–2
M-1.2 Error handling M–3
M-1.3 Procedures in File_IO_Lib M–4
M-1.3.1 REGISTERED M–4
M-1.3.2 REGISTER M–4
M-1.3.3 UNREGISTER M–5
M-1.3.4 OPEN M–5
M-1.3.5 CLOSE M–6
M-1.3.6 DELETE M–6
M-1.3.7 HANDLE_VALID M–6
M-1.3.8 END_OF_LINE M–7
M-1.3.9 END_OF_FILE M–7
M-1.3.10 PUT M–7
M-1.3.11 GET M–8
M-1.3.12 SKIP M–8
M-1.3.13 NEW_LINE M–8
M-1.3.14 PUT_LINE M–9
M-1.3.15 GET_LINE M–9
M-1.3.16 SKIP_LINE M–9
M-1.3.17 EXECUTE M–10
M-1.3.18 ADD_FILE_TO_TEST_SESSION M–10
M-2 Installation of SAS_FILE_IO and FILE_IO_LIB M–11

N MDB CONSISTENCY CHECKS N–1
N-1 Consistency Checker – List of CGS Standard Consistency Checks N–1
N-1.1 Mandatory Checks N–1
N-1.2 Uniqueness Checks N–1
N-1.3 Referential Integrity Checks N–1
N-1.4 Cross Reference Checks N–2
N-1.5 Check Minimum Number of Records N–2
N-1.6 Double SID Check N–2
N-1.7 Privileges Check N–3
N-1.8 Nickname Check N–3
N-1.9 CGS Aggrgate and End Item Type related Special Checks N–4
N-2 List of Single Enditem Checks (Check MDB Item) N–36

O CIS INTERFACE DESCRIPTION N–1

Copyright per DIN 34

FIGURES

Figure 3–1 : CGS and SAS Communication Architecture 3–4
Figure 3–2 : Symbolic Representation of CGS Buiding Blocks 3–8
Figure 3–3 : Anticipated Flow of Checkout Operation Activities 3–10
Figure 3–4 : The Standalone Configuration 3–12
Figure 3–5 : CGS Test Software Distribution to Nodes 3–12
Figure 3–6 : Possible Hardware/Software Configuration for a SCOE (Test node) 3–14
Figure 3–7 : General Structure of a Checkout System 3–16
Figure 3–8 : Data is Stored in the TRDB for Later Evaluation 3–19
Figure 4–1 : The CGS Start Center 4–2
Figure 4–2 : The Message Window 4–3
Figure 5–1 : CGS Pathname Description 5–3
Figure 5–2 : The I_MDB Navigator Window 5–5
Figure 5–3 : I_MDB: CDU Versions Box 5–7
Figure 5–4 : I_MDB: CCU Versions 5–9
Figure 5–5 : Create CDU Version Box 5–12
Figure 5–6 : Create User Tree Node Box 5–12
Figure 5–7 : Create CCU Version Box 5–15
Figure 5–8 : Node Type List Help 5–18
Figure 5–9 : Exemplary Shown End Item Operations with the I_MDB Navigator Window 5–20
Figure 5–10 : Operations on End Items of Type Automated Procedure 5–24
Figure 5–11 : CLS Editor with UCL Compiler 5–25
Figure 5–12 : CLS Editor: Error Menu 5–26
Figure 5–13 : CLS Editor: Listing Window 5–27
Figure 5–14 : CLS Editor: Confirm the Store Operation 5–29
Figure 5–15 : CLS Editor: The Item Information Window 5–30
Figure 5–16 : CLS Editor: Quit — Confirmation Request 5–31
Figure 5–17 : CLS Editor: Processing a User Library 5–32
Figure 5–18 : CLS Editor: Processing a User Library 5–32
Figure 5–19 : CLS Editor Asking for Confirmation of File Creation 5–34
Figure 5–20 : HLCL Command Sequences in the Unix File System 5–35
Figure 5–21 : Parameter Option Combinations for Various Compilation Units 5–39
Figure 6–1 : The SYSTEM_TOPOLOGY_TABLE 6–2
Figure 6–2 : Each Entry in the Node List can be Defined by Three Attributes – Node Type, Logical Name and CGS Internal Name. 6–4
Figure 6–3 : The Node Type Defines the Function of the Node in the Test Configuration . . 6–5
Figure 6–4 : The Node Type Describes the Function of the Node in the Test Configuration 6–5
Figure 6–5 : The Computer with the Logical Name DBS_01 is Linked to the Node Definition 6–6

Copyright per DIN 34

Figure 6–6 : The Prefix is an Internal Marker 6–6
Figure 6–7 : To Describe a Test Configuration all Information Must be Supplied Here . . . 6–7
Figure 6–8 : The Database and Simulator Nodes are Identified by their Pathname 6–8
Figure 6–9 : The Workstation Node List 6–8
Figure 6–10 : The Workstation Node is Participating 6–9
Figure 6–11 : How To Create a Testnode in the Test Node Definition Window 6–10
Figure 6–12 : The Completed Test Nodes Window. 6–12
Figure 6–13 : CDU List – Five CDUs are Assigned to Test Node TEST_NODE_01 6–12
Figure 6–14 : The House Keeping Values Defined for TEST_NODE_01 are Assigned to the Test Node . 6–13
Figure 6–15 : The SAS is Assigned to a Test Node 6–14
Figure 6–16 : The Housekeeping Identifier is a Number from the List of HK Data 6–15
Figure 6–17 : Ground Library Loaded into CLS Editor 6–16
Figure 6–18 : The CCU EURECA_DEMO References all CDUs which Contents is Needed for the Test . 6–18
Figure 6–19 : I_MDB Navigator Provides the Option to Start the Scoe File Generation Process 6–19
Figure 6–20 : Output of Start_load_scoe in Console Window 6–20
Figure 6–21 : The Overall Concept of TES SAS together with CGS 6–32
Figure 6–22 : The Overall Concept of TEV SAS together with CGS 6–32
Figure 6–23 : Ada Library Structure for SAS Development – this picture shows the visibility rule, not a real directory structure 6–34
Figure 6–24 : Relationship of SAS State Transitions 6–42
Figure 6–25 : Starting the GWDU Editor from I_MDB Panel 6–45
Figure 6–26 : MDE Graphical Language Lexical Elements 6–47
Figure 6–27 : A number of SFBs connected together 6–51
Figure 6–28 : An example of an asynchronous group triggered by a synchronous function block 6–52
Figure 6–29 : Use of two synchronous function blocks to reduce system load 6–53
Figure 6–30 : Asynchronuous chain with FSP and activation never 6–53
Figure 6–31 : A model with external interface (one model input and one model output). . . . 6–54
Figure 6–32 : MDE–GL Atomic Function Block with AIL Implementation 6–56
Figure 6–33 : Sample outline of a decision table 6–66
Figure 6–34 : Mission Configuration Start from CGS Task Selector 6–67
Figure 6–35 : The Create user tree node window 6–69
Figure 6–36 : The Node type list help window 6–69
Figure 6–37 : The CDU with a list of models 6–70
Figure 6–38 : The CSS scope check window 6–71
Figure 6–39 : The CSS scope check window reports missing references 6–71
Figure 6–40 : The CSS decision window 6–72
Figure 6–41 : DBB Master Window with selected model 6–73

Copyright per DIN 34

Figure 6–42 : The Composite Editor User Interface 6–80
Figure 6–43 : The tool buttons of the Composite Editor 6–82
Figure 6–44 : Scrolling in overview mode 6–83
Figure 6–45 : The Composite Editors basic menu 6–84
Figure 6–46 : A Composite Editor showing some block objects. The last created asynchronous function block is selected. 6–85
Figure 6–47 : A Composite Editor showing some I/O items attached to block objects. The last created output is selected. 6–87
Figure 6–48 : The output of function block ASYNC will be connected with the INPUT of COMP. . . . 6–92
Figure 6–49 : The stimulus LONG_REAL is selected in the database 6–93
Figure 6–50 : Moving a splitted connection line 6–96
Figure 6–51 : A Composite Editor showing an example of a logical grouping. Note that ASYNC\OUT1 is connected to COMP\INPUT via logical grouping (signal SIG1) 6–98
Figure 6–52 : A Composite Editor showing an example of a global symbol. Note that SYNC\OUTPUT is connected to ASYNC\INPUT via global symbol DEF. 6–100
Figure 6–53 : A list of all available data types 6–102
Figure 6–54 : The empty state code definition window 6–103
Figure 6–55 : The state code input window 6–104
Figure 6–56 : The initial value selection window for state code variables 6–104
Figure 6–57 : Definition window for a variable of type vector 6–104
Figure 6–58 : Definition window for a variable of type vector with predefined list of data types. 6–105
Figure 6–59 : Definition window for a variable of type matrix with predefined rows and columns. . . . 6–105
Figure 6–60 : Definition window for a variable of type record. 6–105
Figure 6–61 : The Rule Check Parameters window 6–110
Figure 6–62 : The Compilation Parameters window 6–111
Figure 6–63 : Composite Inspector User Interface 6–114
Figure 6–64 : Composite Interface Editor user interface 6–115
Figure 6–65 : Components of the Atomic Editor 6–117
Figure 6–66 : AIL Editor user interface 6–120
Figure 6–67 : Decision Table Editor user interface 6–124
Figure 6–68 : Structure of a decision table 6–125
Figure 6–69 : A decision table showing three different ways to fill the action fields. 6–126
Figure 6–70 : Local variables definition in a decision table 6–127
Figure 6–71 : Macro definitions in a decision table 6–128
Figure 6–72 : Logical AND gate implemented by decision table (2 examples) 6–129
Figure 6–73 : Icon Editor user interface 6–131
Figure 6–74 : The Tree Browser user interface 6–133
Figure 6–75 : Printing the contents of the atomic FBs COUNT_1 to COUNT_4 6–135

Copyright per DIN 34

Figure 6–76 : The cover page of the model document 6–136
Figure 6–77 : This grey rectangle shows the model hierarchy after printing 6–137
Figure 6–78 : The Onboard References Adaptation window 6–139
Figure 6–79 : The Simulation table TABLE_1 is selected in the DBB window for editing. . . 6–140
Figure 6–80 : The Simulation Table editor window (overview mode) 6–141
Figure 6–81 : The Monitoring Parameters window 6–143
Figure 6–82 : The graphical representation elements 6–145
Figure 6–83 : The gauge element scaling parameters 6–145
Figure 6–84 : Monitoring I/O items 6–147
Figure 6–85 : The outputs VALUE_2, VALUE_3 are logged 6–148
Figure 6–86 : The function blocks marked for tracing 6–151
Figure 6–87 : The CDU contains several models 6–186
Figure 6–88 : Select MOCS in the CSS scope window to start the model execution 6–187
Figure 6–89 : The initial CSS Simulation Controller window 6–188
Figure 6–90 : The Connected Users window shows a list of connected users and the status . 6–189
Figure 6–91 : The pending commands window shows the commands and the affected model items . . . 6–190
Figure 6–92 : The Start Simulator window allows to specify the host machine 6–190
Figure 6–93 : Select the kernel from the list of running simulators 6–191
Figure 6–94 : The selected user can receive privileges from the session owner 6–192
Figure 6–95 : The user css_test on ada_s sends a request for the logging privilege 6–192
Figure 6–96 : The user css_test on ada_s received the logging privilege. 6–192
Figure 6–97 : The broadcast message is send to all connected users 6–193
Figure 6–98 : The CSS Simulation Controller window with running Simulator 6–194
Figure 6–99 : Select the simulation state from a list 6–195
Figure 6–100 : Setting the simulation mode parameters 6–196
Figure 6–101 : Select a simulation table from the list of available tables 6–197
Figure 6–102 :The Simulation Controller window with two active Session Observer windows showing different levels of the model 6–198
Figure 6–103 : The Session Observer used to oberve the simulation 6–199
Figure 6–104 : Snapshot values are displayed in the MOCS Console window 6–201
Figure 6–105 : The Assign window 6–201
Figure 6–106 : Specifying a time tag in SMT 6–202
Figure 6–107 : Specifying a duration in SMT 6–202
Figure 6–108 : The confirmation window certifies the submission of a time tagged assign command . . 6–203

Copyright per DIN 34

Figure 6–109 : The SMT input window 6–203
Figure 6–110 :The SMT increment input window 6–204
Figure 6–111 :The Simulation Mode window in default state 6–205
Figure 6–112 :Monitoring the same parameter shows different values 6–206
Figure 6–113 :The ICP window is used for command sequence testing 6–208
Figure 6–114 :The flags can be set interactively 6–209
Figure 6–115 :A part of the file .....ARCH_TXT 6–210
Figure 6–116 :The file ...LOG_TXT contains a list of runtime IDs 6–210
Figure 6–117 :CCSDS Primary and Secondary Header 6–241
Figure 7–1 : Example of a Set–Up with three Active Test Configurations 7–8
Figure 7–2 : Test Node State Transitions 7–9
Figure 7–3 : Test Session State Transitions 7–10
Figure 7–4 : 7–20
Figure 7–5 : The TSCV Main Window 7–20
Figure 7–6 : Confirm Shutdown 7–22
Figure 7–7 : The NTP Status Check Window 7–23
Figure 7–8 : Confirm Unload 7–26
Figure 7–9 : Close Session 7–27
Figure 7–10 : The Load Test Configuration for Online Test Window. 7–33
Figure 7–11 : The Select CCU Window 7–36
Figure 7–12 : The Create Test Session Window 7–39
Figure 7–13 : The Maintain Test Session Window 7–42
Figure 7–14 : Delete Data in Default Test Session Notice Window 7–47
Figure 7–15 : The Node Property Sheet Window 7–48
Figure 7–16 : The Replay Properties Window 7–50
Figure 7–17 : The Maintain System Topology Window 7–54
Figure 7–18 : The Display Request Window 7–57
Figure 7–19 : Invocation of Online Control with Example CGS Start Center 7–69
Figure 7–20 : Online Control (HCI) 7–70
Figure 7–21 : Execution Nodes Submenu Selection 7–71
Figure 7–22 : HLCL Commanding 7–72
Figure 7–23 : History Window 7–73
Figure 7–24 : Flags Menu 7–75
Figure 7–25 : Find Dialog 7–76
Figure 7–26 : ADU Status Window 7–78
Figure 7–27 : AP Status Window 7–79
Figure 7–28 : Clock 7–79
Figure 7–29 : Clock (SMT not available) 7–80

Copyright per DIN 34

Figure 7–30 : Clock in Replay Mode 7–80
Figure 7–31 : Database Node Status Window 7–81
Figure 7–32 : Out of Limit Display 7–83
Figure 7–33 : Out of Limit Display Preferences 7–84
Figure 7–34 : Monitoring Window 7–86
Figure 7–35 : Monitoring Window Save Configuration 7–87
Figure 7–36 : Monitoring Window Load Configuration 7–88
Figure 7–37 : Monitoring Window Add Items Dialog 7–89
Figure 7–38 : Monitoring Window" Add in progess...." 7–89
Figure 7–39 : Monitoring Window Preferences 7–90
Figure 7–40 : Raw Data Dump 7–93
Figure 7–41 : Raw Data Dump Window (Load Packet) 7–93
Figure 7–42 : Raw Data Dump Window (Full Mode) 7–94
Figure 7–43 : SAS Status Window 7–95
Figure 7–44 : System Advisory 7–96
Figure 7–45 : EGSE Test Nodes Window 7–97
Figure 7–46 : Explorer Window 7–99
Figure 7–47 : Execution Node Status – General 7–101
Figure 7–48 : Execution Node Status – Data Generation 7–103
Figure 7–49 : Execution Node Status – Links 7–104
Figure 7–50 : Execution Node Status – Monitoring 7–106
Figure 7–51 : Execution Node Status – Time 7–107
Figure 7–52 : I_MDB Tool Properties Showing CM Status 7–109
Figure 7–53 : Convert all GWDU Displays of a CCU to USS 7–110
Figure 7–54 : GWDU to USS Display Conversion Dialog 7–111
Figure 7–55 : XML SCOE Files Generation 7–111
Figure 7–56 : Generating XML SCOE Files 7–112
Figure 7–57 : Conversion Error 7–112
Figure 7–58 : GWDU Batch XML Report 7–113
Figure 7–59 : Start USS Editor 7–114
Figure 7–60 : USS Editor – XML SCOE File Selection 7–115
Figure 7–61 : USS Editor – Export to MDB 7–116
Figure 7–62 : IMDB – Import USS Displays 7–117
Figure 7–63 : USS Display Importer 7–117
Figure 7–64 : Browse File System to Select Root Directory 7–118
Figure 7–65 : Importing USS Displays 7–118
Figure 7–66 : Importing Updated USS Displays 7–119
Figure 7–67 : Screen Setup Maintenance 7–122
Figure 7–68 : Screen Setup – CCU Conflict 7–123

Copyright per DIN 34

Figure 7–69 : Rename Screen Setup Dialog 7–123
Figure 7–70 : UCL Browser Window Example 7–124
Figure 7–71 : UCL Debugger 7–125
Figure 7–72 : Quitt Online Control 7–126
Figure 7–73 : Input Dialog 7–127
Figure 7–74 : User Defined Window Footer 7–128
Figure 7–75 : Concept of End Item Monitoring 7–132
Figure 7–76 : GDU Processing Overview 7–136
Figure 7–77 : GDU Verification 7–139
Figure 8–1 : TEV Main Window 8–6
Figure 8–2 : The TEV Sessions Menu 8–7
Figure 8–3 : Initialize Evaluation Session Window 8–8
Figure 8–4 : Evaluation Session Size Window 8–9
Figure 8–5 : Select Execution Sessions 8–12
Figure 8–6 : Select Test Sessions Window 8–14
Figure 8–7 : TEV : Selected Execution Sessions Window 8–16
Figure 8–8 : TEV : CCU List in MDB Window 8–17
Figure 8–9 : TEV: CCUs From Default Execution Session Window 8–18
Figure 8–10 : Export/Import Tool Window 8–19
Figure 8–11 : The Different Evaluation Tools (note: in CGS 6.3.2, the menu items Statistics Generation... and Graph Display... are available only in backward compatibility mode) 8–21
Figure 8–12 : Definition Menu and its Pop–Up Menu 8–22
Figure 8–13 : Selecting a Time Frame from an Execution Session 8–24
Figure 8–14 : The Events Logging Tool Main Window 8–27
Figure 8–15 : Event Logging Window 8–29
Figure 8–16 : Result Menu and the List of Result Files 8–31
Figure 8–17 : Start of the Raw Data Dump Tool 8–33
Figure 8–18 : Raw Data Dump Window 8–34
Figure 8–19 : Select the ADUs and GDUs from the Mission Database 8–38
Figure 8–20 : Select the ADUs and GDUs from the Archive 8–38
Figure 8–21 : Packet Navigator window showing a raw data packet 8–40
Figure 8–22 : Output Format of a Raw Data Dump 8–41
Figure 8–23 : A Summary Format 8–42
Figure 8–24 : Start of the Data Set Generation Tool 8–44
Figure 8–25 : TEV: Data Set Generation Window 8–45
Figure 8–26 : Time Based Sampling Semantics 8–46
Figure 8–27 : Selection of Parameters Below the Virtual Node Name Tree 8–50
Figure 8–28 : Start of the Statistics Generation Tool 8–52
Figure 8–29 : The Statistics Generation Tool Window 8–53

Copyright per DIN 34

Figure 8–30 : Start of the Data Listing Tool 8–55
Figure 8–31 : TEV Data Listing window 8–56
Figure 8–32 : The Data Listing Generation Tool Window 8–59
Figure 8–33 : Start the Graph Tool 8–60
Figure 8–34 : The Graph Tool Window 8–60
Figure 8–35 : Line Graph Properties Window 8–63
Figure 8–36 : Line Graph Properties Window with Colour Selection 8–64
Figure 8–37 : X/Y Graph Properties 8–65
Figure 8–38 : Pie Chart Properties 8–66
Figure 8–39 : Examples of Line Graph and Bar Chart 8–66
Figure 8–40 : Example of XY Graph and Pie Chart 8–67
Figure 8–41 : Utilties Menu 8–68
Figure 8–42 : The Working Directory Files Manager 8–69
Figure 8–43 : The Command Window to Copy a File plus Confirmation Window 8–70
Figure 8–44 : Start the Data Set/Events Merger 8–71
Figure 8–45 : When No Session Selected 8–72
Figure 8–46 : No Session Selected : Time Frame Combination 8–73
Figure 8–47 : Session Selected with Selected Time Frame from Events Listing 8–74
Figure 9–1 : The Main Menu of the Recovery Scripts 9–75
Figure 9–2 : Selection of the Execution Session Menu from the Main Menu 9–76
Figure 9–3 : Execution Session DataDiagnostic Report. 9–79
Figure 9–4 : Selection of the ’Delete Execution Session’ Menu 9–79
Figure 9–5 : Selection of the ’Delete Default Test Session Data’ Operation. 9–80
Figure 9–6 : Selection of ’Delete the On–Line Data of an Archived Session’ Operation. . . 9–81
Figure 9–7 : Selection of the ’Delete Completely a Test Execution Session’ Operation. . . . 9–82
Figure 9–8 : Import from FA Failure, Execution Session. 9–83
Figure 9–9 : Export on FA Failure, Execution Session. 9–84
Figure 9–10 : Archiving on FA Failure, Execution Session. 9–85
Figure 9–11 : Retrieving from FA Failure, Execution Session. 9–86
Figure 9–12 : Deletion Failure, Execution Session. 9–87
Figure 9–13 : File Storage Failure, Execution Session. 9–88
Figure 9–14 : List Local Files Unstored, Execution Session. 9–89
Figure 9–15 : Remove Local Files Unstored, Execution Session. 9–90
Figure 9–16 : Close Execution Session. 9–91
Figure 9–17 : Execution Session List. 9–92
Figure 9–18 : File Transfer Failures. 9–93
Figure 9–19 : Recover Files for Multiple Sessions with Auto–Detection 9–94
Figure 9–20 : Recover Archive Files for one Test Node 9–96
Figure 9–21 : Recover Event/Evl Files for one Test Node 9–96

Copyright per DIN 34

Figure 9–22 : Set Aborted Session to Normally Closed 9–98
Figure 9–23 : Selection of the Evaluation Session Menu from the Main Menu 9–99
Figure 9–24 : Report of the Evaluation Session Data Diagnostic 9–101
Figure 9–25 : Selection of ’Delete the On–Line Data of an Archived Session’, Evaluation Session. . . . 9–102
Figure 9–26 : Selection of ’Delete Completely an Evaluation Session’ Operation. 9–103
Figure 9–27 : Import from FA Failure, Evaluation Session. 9–104
Figure 9–28 : Export on FA Failure, Evaluation Session. 9–105
Figure 9–29 : Archiving on FA Failure, Evaluation Session. 9–106
Figure 9–30 : Retrieving from FA Failure, Evaluation Session, no NEW_NAME. 9–107
Figure 9–31 : Retrieving from FA Failure, Evaluation Session, NEW_NAME not null. . . . 9–109
Figure 9–32 : Deletion Failure, Evaluation Session. 9–109
Figure 9–33 : File Storage Failure, Evaluation Session. 9–110
Figure 9–34 : List Local Files Unstored, Evaluation Session. 9–111
Figure 9–35 : Remove Local Files Unstored, Evaluation Session. 9–112
Figure 9–36 : Remove Evaluation Users Menu. 9–113
Figure 9–37 : List the Connection References of Evaluation Applications 9–114
Figure 9–38 : Delete the Connection References of an Evaluation Application 9–115
Figure 9–39 : List the Allocation References of all Evaluation Users 9–116
Figure 9–40 : Delete the Allocations of an Evaluation User 9–117
Figure 9–41 : Evaluation Session List. 9–118
Figure 9–42 : DBS Error Number Explanations 9–120
Figure 9–43 : DBS Error Number Explanations 9–120
Figure 10–1 : Example GUI for End Item Pirvilege Allocation 10–9
Figure 10–2 : Example GUI on Library Privilege Allocation 10–10
Figure 1 : Typical ORACLE Error Message D–1
Figure 2 : ORACLE On–Line Error Help D–1
Figure 3 : Oracle Error Number Added to an I_MDB Specific Error Message D–2
Figure 4 : The UCL Compiler Error Message is displayed in the lower part of the editor window. . D–5
Figure 5 : The On–Line Syntax Help shows the UCL syntax D–5
Figure 6 : A Syntax Error is Indicated by a Dark Rectangle, the Error Message is Displayed in the Lower Part of the Editor Window. D–6
Figure 7 : Error message displayed in the Composite Editor message line D–7
Figure 8 : Error message displayed in the Atomic Editor message line D–8
Figure 9 : The CSS MDE rule check window D–8
Figure 10 : Transcript window contents – First part D–8
Figure 11 : Transcript window – Second part D–9
Figure 12 : Transcript window contents – Third part D–10
Figure 13 : Transcript window contents – Fourth part D–10

Copyright per DIN 34

Figure 14 : The Internal Error window D–22
Figure 15 : Syntax errors in the HLCL command are shown by a little arrow. D–97
Figure 16 : Errors during command execution D–97
Figure 17 : MOCS message window D–99
Figure 18 : A concatenated error string is displayed in the ICP window D–99
Figure 19 : Error Report example in TEV D–138
Figure 20 : Different Types of HLCL Commands H–1

Copyright per DIN 34

TABLES

Table 5–1 : List of General Operations on Nodes including End Items 5–19
Table 6–1 : Housekeeping Data Description 6–25
Table 6–2 : Housekeeping Data Description (continued) 6–28
Table 6–3 : Housekeeping Data Description (continued) 6–30
Table 6–4 : Restrictions in the ADA X–VIEW Environment 6–36
Table 6–5 : Basic MDA Data Types, Value Ranges and Mapping to ORACLE Data Types 6–214
Table 6–6 : Basic MDB Enumerations and Values 6–215
Table 10–1 : CGS System File Structure 10–3
Table 10–2 : CGS User File System Structure 10–4
Table 10–3 : CGS privileges 10–6
Table 10–4 : MDB privileges 10–6
Table 10–5 : CIS privileges 10–6
Table 10–6 : mapping system privileges – old user roles 10–6
Table 10–7 : Subcommands for the PRIV Tool 10–8

Copyright per DIN 34

1 INTRODUCTION

1.1 Identification and Scope

This document is the CGS User Manual as specified in chapter 2.1.

1.2 Purpose

This Manual provides the CGS user with a top–level introduction how to use CGS for software development, mission preparation and system test purposes.

1.3 How this document is organised

This Manual has been organised to provide the user the following:

– an overview of the concept behind the major CGS functions
– an introduction how to use the major CGS functions

For detailed information on the use of some CGS tools the user will have to refer to the lower level CGS component Reference Manuals and User Manuals identified in Chapter 2 'Documents' of this Manual. This is particularly the case with the Software Development Environment (SDE) and the Mission Database Application (MDA) and with the Window Definition Utilities (FWDU and GWDU). In these cases this Manual provides the user with information regarding the top level CGS application of such tools.

Chapter 1	**Introduction** *Provides a general overview of the Manual*
Chapter 2	**Documents** *Lists all referenced documentation*
Chapter 3	**CGS – The Development and Test Support System** *Describes the CGS functions, hardware/software environment, CGS concept and available CGS configurations for each user task*
Chapter 4	**CGS General Setup Tasks** *Describes how to setup the CGS user environment, how to start CGS and an overview of the error message handling concept*
Chapter 5	**Development Support Services** *Describes how to use the CGS ADA/C programming programming environments and how to use the documentation preparation tool*
Chapter 6	**Mission Preparation Tasks** *Describes how to build Mission Configurations with CGS and how to prepare the contents of the data forming part of a mission configuration*
Chapter 7	**Test Preparation** *Describes how to setup a test configuration, develop APs, SASs, ground synoptic displays and simulation models*

Copyright per DIN 34

Chapter 8	**Integration and Test** *Describes how to setup the test environment with CGS and how to execute a test session*
Chapter 9	**Test Evaluation** *Describes how to use CGS to evaluate and report on the results of a test execution session*
Chapter 10	**TRDB Tools** *Describes how to use CGS to maintain the Test Result Data base (TRDB)*
Appendices	*Provides detailed information with respect to end items, error messages, configuration parameter, SAS API and test–related HLCL commands*

Copyright per DIN 34

2 DOCUMENTS

2.1 Applicable Documents

2.1.1 CGS–RIBRE–SPE–0001
CGS V6 Specification
Issue 2/D , 23.03.2004

2.1.2 CGS–RIBRE–ICD–0001
System to CGS Interface Control Document, Issue 1/–, 31.01.2002

2.1.3 CGS–RIBRE–IRN–CGS–8332
System to CGS Interface Control Document, Issue 1/B

2.2 CGS User Manuals

The following manuals are part of the CGS User Manual delivery and build together with this manual the complete CGS User Manual documentation set.

2.2.1 MDA Manuals

2.2.1.1 Deleted (Covered by 2.2.1.2, 2.2.1.3, 2.2.1.4)

2.2.1.2 COL–RIBRE–MA–0030–00
MDA Introduction Manual
Issue 3/B, 04.04.1997

2.2.1.3 COL–RIBRE–SUM–0003–00
MDA Reference Manual
Issue 1/–, 09.11.2001

2.2.1.4 COL–RIBRE–MA–0018–00
MDA Administration Manual
Issue 4/B, 31.03.2000

2.2.1.5 Deleted (covers documents 2.2.1.6, 2.2.1.7 and 2.2.1.8)

2.2.1.6 COL–RIBRE–MA–0037–00
DADI–MA Introduction Manual
Issue 3/–, 04.04.1997

2.2.1.7 CGS–RIBRE–SUM–0005
DADI–MA Reference Manual
Issue 1/–, 09.11.2001

2.2.1.8 CGS–RIBRE–SUM–0006
DADI–MA Administration Manual
Issue 1/–, 09.11.2001

Copyright per DIN 34

2.2.1.9 COL–RIBRE–MA–0046–00
SID Range Tool Users and Operations Manual
Issue 1/–, 15.09.1997

2.2.1.10 CGS–RIBRE–MA–0001
UCL Debugger User Manual
Issue 1/–, 01.09.2004 (planned)

2.2.2 WDU Manuals

2.2.2.1 UM–114–001–ROV
GWDU User's Manual and Operations Manual
Issue 1.5, ???

Copyright per DIN 34

2.3 Referenced User Manuals

2.3.1 CGS–RIBRE–MA–0007
Start Center – A generic user interface for multi–process software systems
Issue 1/–, 01.08.2006

2.4 Other Referenced Documents

2.4.1 CGS–RIBRE–SUM–0002
CGS Installation Manual
Issue 2/A, 01.03.2004

2.4.2 CGS–RIBRE–ICD–0002
MDB Standard Entities and Application Programming Interface
Issue 1/–, 31.01.2002

2.4.3 CGS–RIBRE–STD–0001
User Control Language (UCL) Reference Manual
Issue 2/–, 25.05.2004

2.4.4 CGS–RIBRE–STD–0002
High Level Command Language (HLCL) Reference Manual
Issue 2/–, 25.05.2004

2.4.5 CGS–RIBRE–STD–0003
UCL Stack Machine and I–Code Reference Manual
Issue 2/–, 25.05.2004

2.4.6 COL–RIBRE–STD–0008
Reference Manual for Crew Procedure Language and Software Commanding
Issue 1/F, 31.10.2001

2.4.7 STD 1216804
Ground Human–Computer Interaction Standards
Issue 5/B, 28.02.1993

2.4.8 Sun Ada Programmer’s Guide, SUN Microsystems
March 1992

2.4.9 ISBN 0–201–52364–7
Open Look Graphical User Interface. Application Style Guidelines

Copyright per DIN 34

2.4.10 ISBN 0–201–52365–5
Open Look Graphical Interface. Functional Specification

2.4.11 The GNU Ada 95 Compiler
Version 3.16a
Document revision level 1.316.2.2
2003/01/21

2.4.12 USS User Manual

Copyright per DIN 34

3 CGS – THE DEVELOPMENT AND TEST SUPPORT SYSTEM

3.1 Overview of CGS

3.1.1 Function and Purpose

The COLUMBUS Ground System (CGS) constitutes the set of products which support or enable various activities performed during Design and Development and the Integration, Test and Qualification of the Flight Configurations. CGS will be utilized in different ground based Facilities of various Space Programmes. Major Facilities currently identified are:

- Software Design and Development Facility ,
- Electrical Ground Support Environment (EGSE),
- Software Integration and Test Environment .

CGS will support the following functionality:

Design and Development Support for Ground and Flight Systems including:

- Columbus Ground System Infrastructure (CGSI) provides the basic service layer to all CGS S/W Applications, supporting Servers (SUN), Test Nodes (SUN) and graphical Workstations (SUN) under the UNIX Operating System (Solaris).
  It provides a Top Level User Interface with menus to startup remaining applications and a message window to present messages from the various tools to the user.
- The *Mission Database Application (MDA)* constitutes the set of utilities which support or enable various activities typically performed during the preparation phase of a checkout / simulation test or mission and provides data entry and reporting, configuration management and support to off-line generation of onboard database/flight image. MDA is centered around the Mission Database which serves as a central repository for all test / mission–related information.
- The *Columbus Language System (CLS)* comprises several language related software components for UCL and HLCL:
  - The CLS Editor is the XView based user front–end for editing UCL or HLCL command sequences as well as for the generation of CPL sequences and parameter list editing and for the specification of SW Commands within the interactive database environment of MDA,
  - The UCL Compiler translates automated procedures (APs) and libraries written in the User Control Language (UCL) into an intermediate code (I–Code), which can be executed by the Test Execution Software (TES),
  - UCL based automated procedure development and maintenance is supported by dedicated UCL debugger.
- The *Ground Window Definition Utility (GWDU)* provides functionality to generate checkout and simulation orientated ground synoptic displays. These layouts contain animated functional drawings which will be used to display e.g. checkout, simulation, subsystem and payload status information and to read in dedicated commands from the human user. This product will be based on the Ground Symbol and Display Standard.

Copyright per DIN 34

Integration, Test and Qualification Support for Ground and Flight Systems:

- The *Test Setup, Configuration and Verification Software (TSCV)* configures a required system configuration for a Checkout Test / Simulation. It enables the test nodes to be activated for a given test. TSCV implements the generation of a test session in the master archive. TSCV further supports the user to identify and control the currently used S/W Configuration.

- The *Online Test Control Software* (OTCS) provides all services related to user input / output on workstations during execution of a Checkout Test / Simulation Session. It provides different interfaces to the window system and the HCI devices (keyboard,mouse,screen) and includes services for synoptic display update, user help and user guidance through test operations. It provides the commanding interface and the message output during online test execution.

  OTCS includes the HLCL Command Window, which constitutes the interactive commanding interface of different applications running on various workstations in the ground system. It interprets and executes interactive commands and automated command sequences written in the High Level Command Language (HLCL),

- The *Test Execution Software (TES)* implements the support for test operations and automatic testing / monitoring of the Unit under Test (UUT). It is driven by the test definitions done by MDA and the configuration setup by TSCV. It provides a generic data and control interface to the UUT and all services required for realtime enditem data processing.

- *Data Base Services (DBS)* provides low level management and access to checkout test related result data items stored in the Test Results Database or files.

- *Test Evaluation Software (TEV)* provides all services to evaluate a.m. data generated and stored during checkout test execution. It provides services for data selection and data presentation. It further implements the final archiving of test results as well as the selective exporting / importing of parts of the Test Result Database.

- *Network Software (NWSW)* provides low level message based interprocess communication as well as file transfer and directory services in a non NFS environment.

- *Timing Services Software (TSS)* provides the synchronization of local computer clocks in a distributed environment with respect to the actual (local) time. In addition, TSS also provide low level SMT distribution, access and handling.

- CORBA based client/server interface service (*implemented as CIS*) . Within a session an external client can request specific services such as HLCL command services, telecommand services, telemetry data acquisition service and event acquisition/dispatching services. The CIS server communicates on the client side with CGS services.

Copyright per DIN 34

3.1.2 CGS Hardware and Commercial Software Environment

It should be noted that CGS is a 'pure' software system. The user has therefore to separately procure the necessary hardware and commercial software environment.

CGS V 6.1.x runs on two types of Unix environments as detailed below.

Note: for an more detailed definition refer to the custumer support services. The relevant web link is: http//cs.space.eads.net/cgs

The system can be used on a single workstation or can be configured in a client/server architecture. The test node requirements are the saame as the server requirements.

3.1.2.1 Sun Server Platform

Server:	SUN SPARC, memory ≥ 256 MB, disk ≥ 2 GB
Workstation:	SUN SPARC ≥ 256 MB, color screen

The required commercial software environment to run CGS is as follows:

For the SUN workstation:	SOLARIS 2.8 Oracle 9.2.0.2 Dataviews 9.9/Run–Time GNAT Ada Compiler 3.16a (if Ada API is used)

3.1.2.2 Linux Server Platform

Server:	Intel X86, memory ≥ 1 GB, disk ≥ 2 GB, clock frequency > 1 Ghz
Workstation:	Intel X86, memory ≥ 1 GB, disk ≥ 2 GB, clock frequency > 1 Ghz

The required commercial software environment to run CGS is as follows:

For the Linux workstation:	Linux RedHat 7.2 Oracle 9.2.0.2 Dataviews 9.9/Run–Time GNAT Ada Compiler 3.16a (if Ada API is used)

3.1.3 CGS Communication with a Test and Control Environment

The CGS core system delegates the communication with the CGS environment to dedicated Special Application Systems (SAS). Commnication services can be established with "units under test" or "units under monitoring and control".

A high level overview is given in Figure 3–1.

Copyright per DIN 34

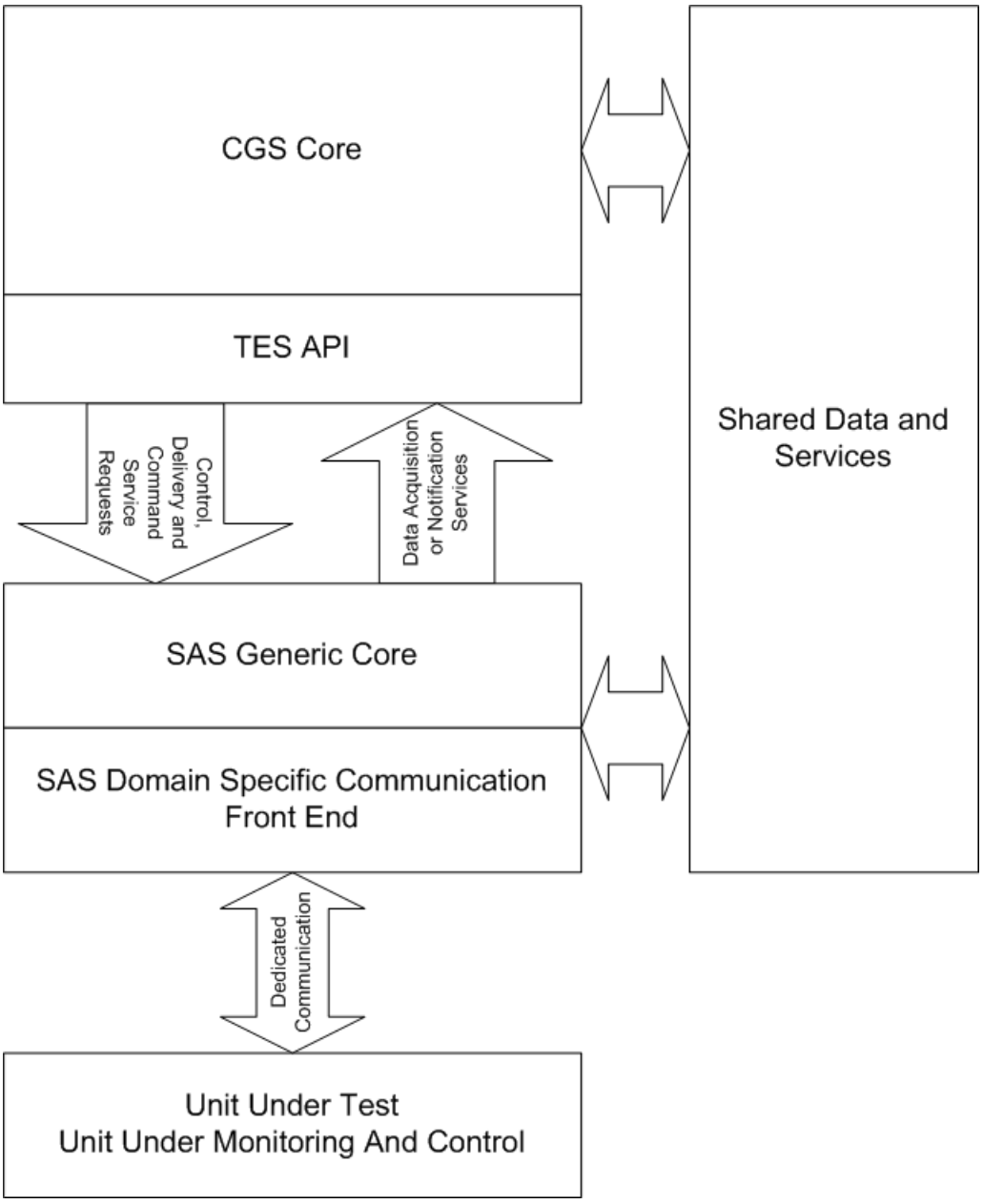


Figure 3–1 : *CGS and SAS Communication Architecture*

CGS core can initiate different service requests to a SAS node such as “get data”, “send stimuli”, etc. The SAS can provide data acquisition and provides other services.

Data can be acquired from the environment by “Acquisition Data Units” (ADU’s) and data can be delivered to domain specific entities by “Generated Data Units” (GDU’s).

Domain specific data transmission protocol services can be translated into CGS requests as they are implemted in the specific SAS communication front end.

When communication to Units Under Control by TM/TC protocols is required, telecommands can be delivered by GDUs to an SAS and telemetry can be acquired from an SAS by ADU’s.

More details are described in chapter 6.2 on page 6–31.

Copyright per DIN 34

Several SAS instances are allowed to communicate with the external environment.

The TES application program interface (API) is described in Appendix F.

- *Project specific SAS implementations shall be part of the project documentation.*

Copyright per DIN 34

3.2 The CGS Checkout and Test System

The purpose of this section is to provide the user with an introduction to the concept of the CGS checkout and test system.

3.2.1 General

Operationally CGS functionality is logically partitioned into two systems:
The **off–line system** supporting the preparation and management of software, hardware and data.
The **on–line system** supporting the integration, checkout and qualification testing of hardware and software.

There are a number of Design Goals for CGS software, involved in the checkout and test operations.

Vertical Commonality

- Where possible, the same test system is used for different test levels eg from Assembly level to System level.
- A test object oriented user interface allows for execution of tests independently of the actual test system configuration, eg an Automated Procedure shall be executed (without change) on different hardware or system configurations.
- Transparent access to the standard services of the system is provided with the use of logical names (”pathnames”) to refer to enditems.

Open Architecture

- Additional Hardware and Software Units can be integrated into a test system, thus meeting different requirements coming from a variety of Users and test objects (UUT: Unit Under Test).

Integrated Toolset

- Integrated tools enable Definition, Execution/Monitoring and Evaluation of a Test.

3.2.2 Basic Concepts, Tasks, and Features

To introduce the user into the terminology as well as to give an overview on the CGS products used for testing, a short description is given in the following section.

The checkout and test system based on CGS products can be used for

- preparation
- configuration
- execution
- evaluation

Test has to be understood as a general term standing for

- integration
- troubleshooting
- checkout

Copyright per DIN 34

- and verification activities

Note that the checkout and test system is sometimes called VICOS (Verification, Integration and Checkout Software) although infact this is a generic name covering a number of tools.

The **Item Under Test** is the hardware or software item being tested (e.g. Ground APs and UUTs) . This may be a ground or flight component.

The **Test Harness** is the framework of software, hardware and data especially produced for testing the Item under Test. It consists of Test Controllers, Test Stubs and Test Data that are needed for the test.

The **Test Controllers** are the software and hardware that control execution of a test. Software test control can range from Ada Tool support to CGS Products: HCI/TES/TSCV. Hardware Test Control can range from a variable resistor to the EGSE Hardware Control System.

Test Stubs are the software and hardware that provide dummy functionality for components not yet available (software 'stubs' and hardware 'dummies'). Software stubs can range from empty software modules through complex Simulation Models to a qualified flight software subsystem. Hardware dummies can range from a short circuit through an EGSE SCOE system to qualified flight hardware subsystem.

CGS Add Ons are required to provide Compilation environments and SW Analysis software, provide interfaces to the Test harness and Test Equipment and also possibly specialist hardware to support development.

System Hardware provides the platform on which both the On–line and Off–line environments execute. For product level production these are usually the same hardware with specialist hardware labelled CGS hardware add–ons. For system testing these platforms are significantly different.

The CGS **Software products** forming the checkout and test system are:

- MDA – the Mission Database Application
- TSCV – the Test System Configuration and Verification Software
- TES – the Test Execution Software
- TEV – the Test Evaluation Software
- DBS – the DataBase Services
- HCI – the Human Computer Interface
- PLATFORM Services:
  CGS Infrastructure (CGSI)
  Network Software (NWSW)
  Time Synchronisation Services (TSS)
- CIS – the CGS Interface Server

Test preparation, as described in the following sections, is mainly the description of the test article (unit under test) as well as the test system (test equipment) in terms of re–usable configuration database enditems.

Copyright per DIN 34

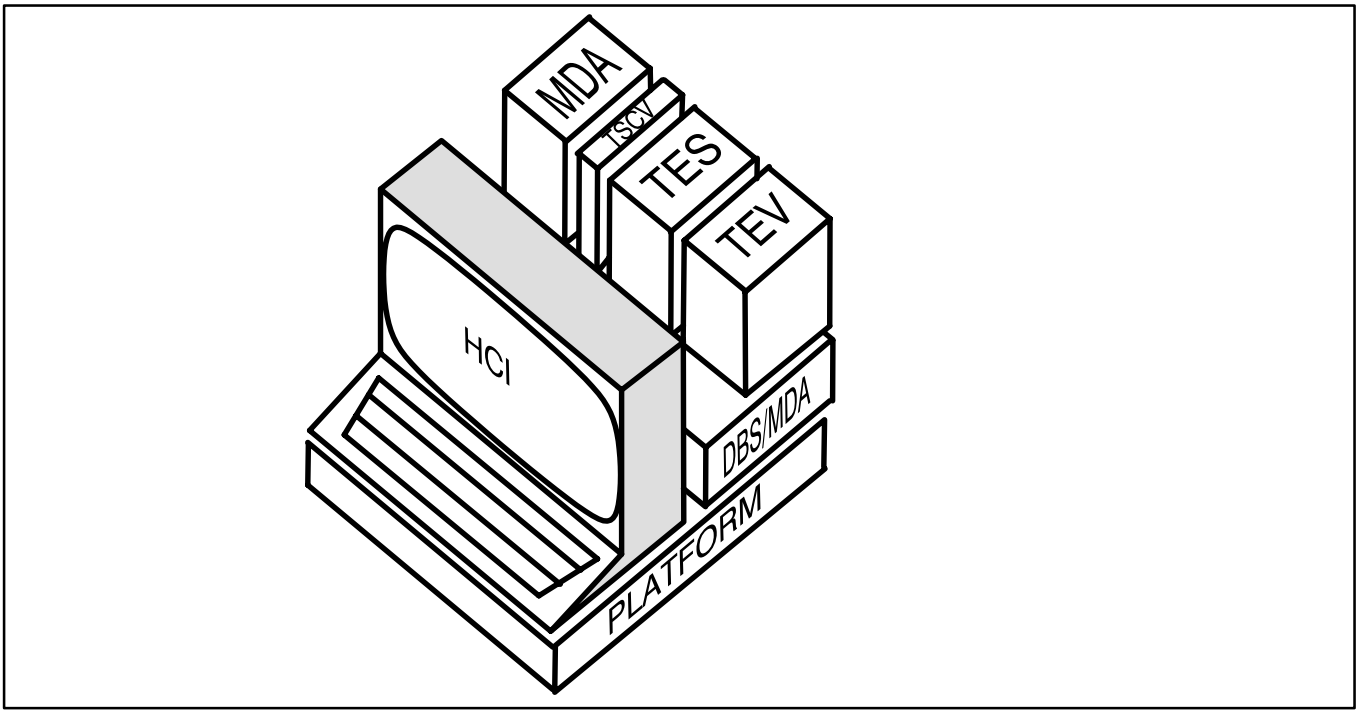


Figure 3–2 : *Symbolic Representation of CGS Buiding Blocks*

The CIS server (not shown in the figure above) communicates with CGS components via TES and other components.

3.2.2.1 Tailoring the Checkout and Test System to a Specific Test System Configuration

The architecture of the unit under test and the test equipment components needs to be described by the concepts of the CGS name tree and CGS end item types.

Re–use of onboard configuration data by proper mapping can be achieved.

Before the software products listed earlier can be used, the following definitions must have been done:

- Definition of UUT end–items and UUT H/W – S/W configurations within the Mission DB (MDB)
- deleted
- Definition of EGSE acquisition and stimuli end–items within the MDB
  This includes for the measurement descriptions:
  – Describe the attributes of a measurement point
  (in the test equipment or the unit under test)
  – Physical acquisition information for hardware devices
  – Calibration from raw to engineering values
  – Monitoring
  – Emergency actions in case of limit violations

  and for the stimuli descriptions:
  – Describe a stimulus inside the test equipment or to the unit under test
  – Configuration information for the hardware device
  – Parameter definition
  – Related actions in the test system

Copyright per DIN 34

- Definition of nodes used in the test configuration
  - Describe the role of (computer) nodes
  - Make nodes accessible via nametree
  - Perform logical to physical name translation
- Definition of the checkout and test system S/W configuration to be loaded to each node (this includes the definition of the test configuration tables specifying the allocation)
  - Describe the version and location of EGSE Software
- Development of Specific Application S/W (additional Ada modules on top of the test system (only needed for a specific test))
- Definition of general UCL and HLCL sequences for EGSE activation/ shutdown/ operational control

3.2.2.2 Preparing a Specific Test

A specific test is be defined in the following way:

- Establishment of a test plan providing information on test requirements, test environments and configuration control of the EGSE and UUT. This will be done with support from the CGS off–line system or manually.
- Development of operational test definitions (shall be performed in the MDB) which includes:
  - Definition of UCL/HLCL sequences
    The user may either edit and compile automatic procedures (AP) written in UCL or define interactive command sequences (HLCL sequence) that allow symbolic operations and sequencing of single commands.
  - Definition of windows/pictures
    The user may define colored graphical representations (pictures) of the UUT and the EGSE and of the actual status of the devices during online operations. The picture can be established by an interactive drawing tool (WDU) that defines graphical symbols of high resolution and their references to end–items defined in the DB.
  - Definition of automatic monitoring
    The user may define automatic monitoring of end–items assigning multiple limit sets to end–items and defining automatic actions to be performed in a (dangerous) out of limit situation. Such actions may be UCL procedures, telecommands or commands to the EGSE H/W (stimuli).
  - Definition which predefined configuration shall be loaded for test
    To enable the test nodes for fast execution of UCL procedures, data formatting etc. the user has to allocate a specific function (e.g. monitoring of a specific UUT sub–system) to a test node. This results later in the load of specific tables (e.g. all tables of a subtree which define the configuration of the resp. SS) from the DB to the test nodes memory.
    Furthermore the user defines the default test system configuration be specifying those nodes that shall take part of the test.

3.2.2.3 Database Driven System

Figure 3–3 shows the way operations are driven by the different databases:

Copyright per DIN 34

Operations are driven by a configuration database which

- contains description of the UUT and the test system.
- contains also the automated test procedures, synoptic picture definitions,etc
- utilizes SDDF version control and user authorisation mechanisms
- data are logically grouped

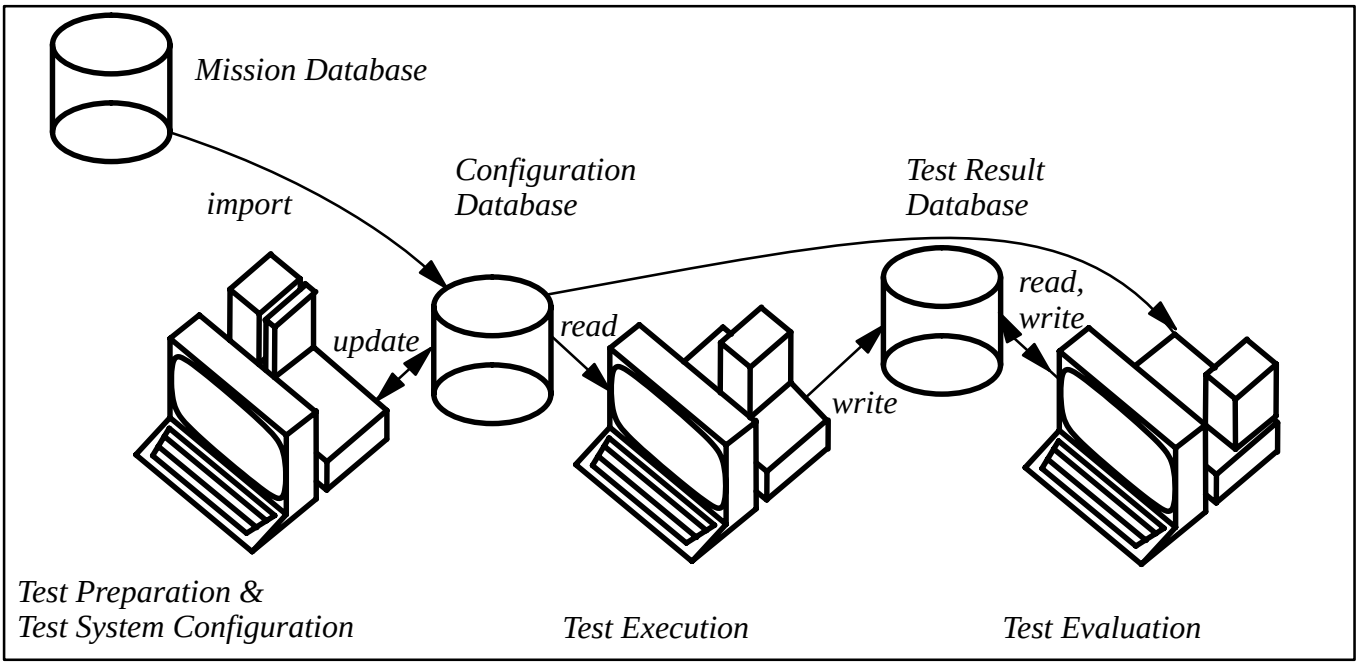


Figure 3–3 : *Anticipated Flow of Checkout Operation Activities*

Test results are stored in the test result database which

- contains test result data
- serves as intermediate storage of test results on magnetic media
- serves as final archive = long term storage of data on other media (currently optical disc)
- serves as Master archive covers both, intermediate and long term storage

3.2.2.4 Distributed Configuration Concept

The CGS software for checkout and test operations is divided into services for Test Session definition, Test Execution/Control and Test Evaluation.

Central Database repositories provide storage and management of all prepared tables, test results and information supporting the integration activities. In particular a derivative of the Mission Database provides a Test Configuration Database.

The tool design is based on the concept of logical nodes which might be mapped to physical processors in a distributed environment.

Copyright per DIN 34

There are 3 types of nodes involved in checkout and test operations.

Workstation Nodes

For a Workstation node, the User Interface functions for controlling the execution of Tests are provided. This includes driving of windows and interactive dialogs. This software is available via the CGS Top Level User Interface in particular the Test Set Up, Test Execution and Test Evaluation tasks. (ie CGS Products TSCV, TES and TEV).

In particular, during Test Execution, messages and dynamic display updates are distributed to and from the Workstation Nodes to and from the execution resources.

Test Nodes

For a Test node, the Standard Services required during test configuration/execution such as monitoring, command handling etc, are implemented by the Test Execution Software (TES) product.

There are two types of test nodes:

- Local Test Nodes that provide only local monitoring
- One Global Test Node providing the overall monitoring of the system (MTP).

CIS can be part of a test node.

Database Server Node

This node provides data services such as DBMS, logging/archiving management, printing service etc. The structure of this node is different from the others as it does not have a service requesting part, but only a service provider part. Note that Test Configuration DB and Test Results DB services are provided by Database Server nodes.

Note that a node is not necessarily a separate computer system !

From the allocation of nodes to processors, the following configurations can be derived:

– the **Standalone Configuration**

This configuration comprises one of each node type, i.e. a Workstation, a Test node and a Server node. These nodes reside on two computers: one executing the test node and the other executing the Workstation node and the DB Server node

Copyright per DIN 34

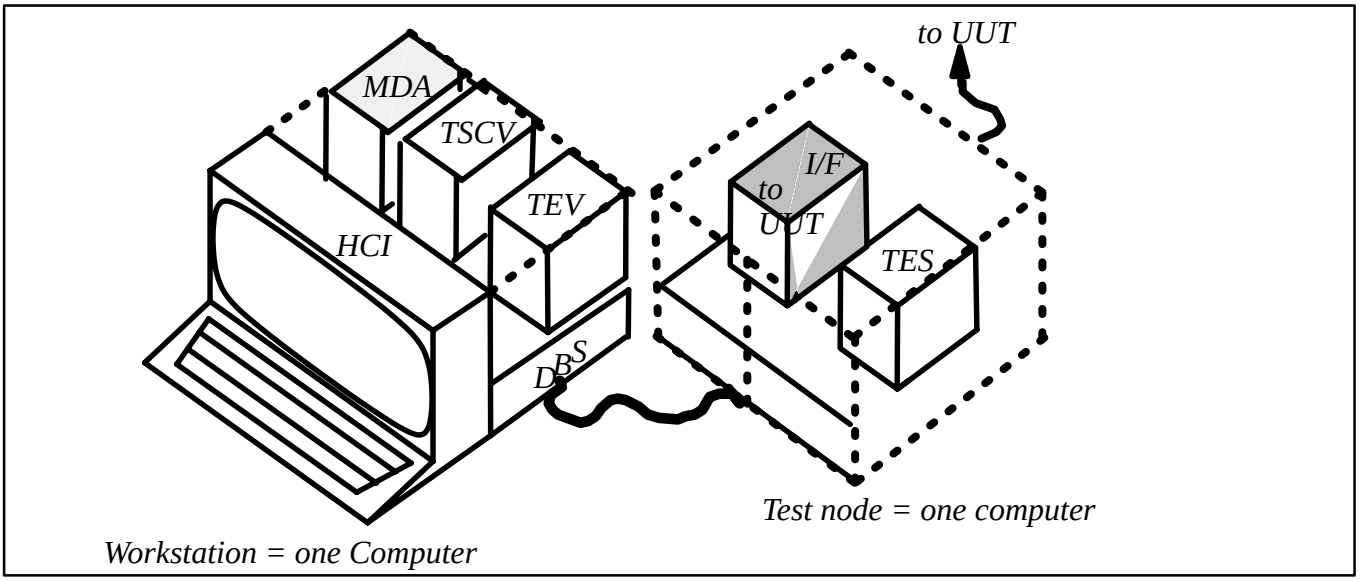


Figure 3–4 : *The Standalone Configuration*

– the **Distributed Configuration**

This configuration includes one DB Server node, several test nodes and several workstations. Each node will then probably reside on one computer, but combining several nodes to one computer is possible.

Note: The combination of two nodes of the same type (e.g. two test nodes) on the same computer is not supported.

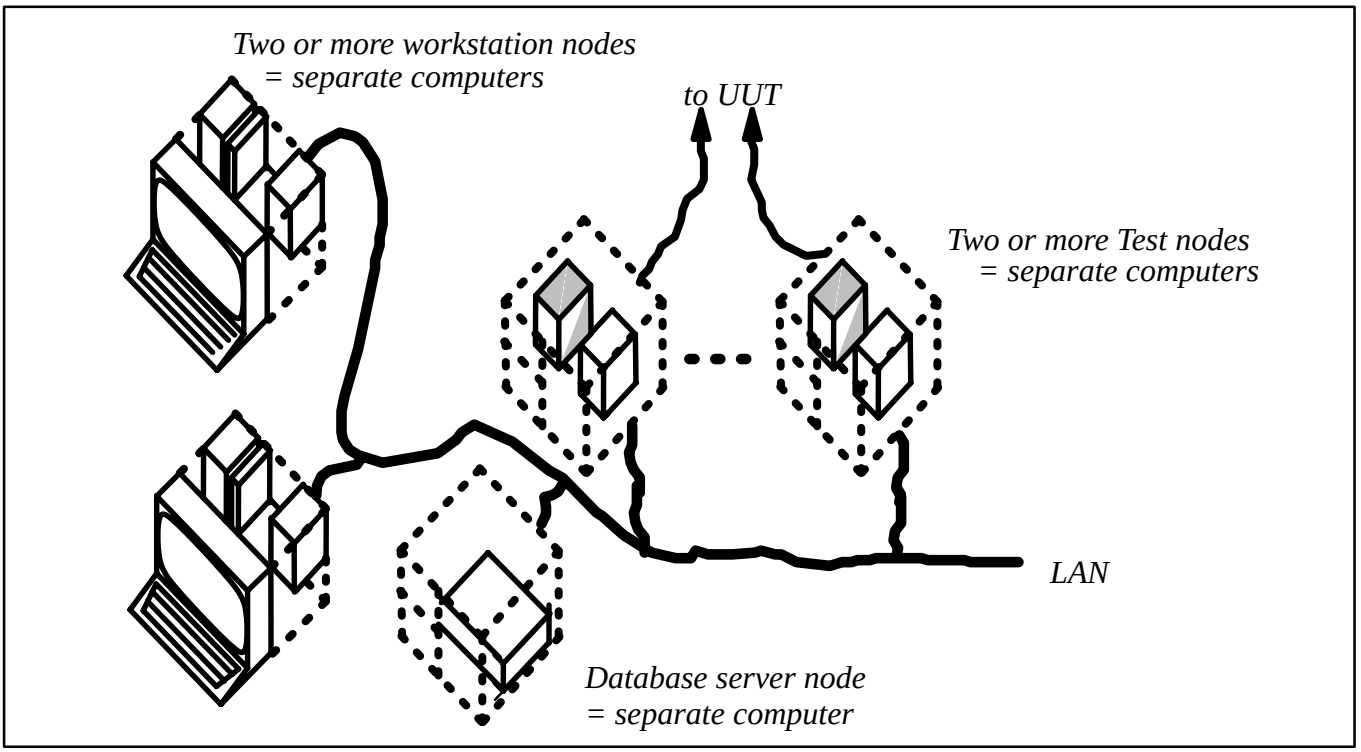


Figure 3–5 : *CGS Test Software Distribution to Nodes*

Copyright per DIN 34

- *A test system comprises a variable number of nodes according to the functional needs of the test – a configuration can comprise 1 to 32 test nodes and 1 to 32 Workstation nodes (On-line).*
- *A test system comprises a variable number of computers according to the performance needs of the test.*

3.2.2.5 The Test System Open Architecture

Among various test objects there will be significant differences in the provided test interfaces. CGS does not aim to include all these interfaces and to implement all the processing required for them.

Instead CGS provides a flexible application software interface that can be used by each User (i.e. SCOE or EGSE manufacturer) to implement the respective interfaces and processing software which runs on the same Node as the CGS software and uses CGS Testing services.

Special Application Software (SAS) constitutes the interface between the checkout and test system and the unit under test (UUT)

- to read data
- to issue stimuli

The checkout and test system provides interfaces for the Special Application Software (SAS) to allow a user to use the monitoring, automatic procedure capabilities and user interface provisions also for data retrieved from or sent to non–standard interfaces.

The figure below shows a possible architecture scenario, indicating possible SAS extenssions.

SAS are separate operating system processes running under CGS control and communicating with CGS via standardised, internal mechanism. The SAS interface is on Ada procedural level.

- *Note that it is a customer task to create the appropriate SAS !*

The special application software allows additional features and allows interfacing of COTS (e.g. EXCEL/ ACCESS).

SAS can read data from the checkout and test system and can provide data to the system for the purpose of special computations:

- to display data in a special way
- to do mathematical transformations
- to store data in a special way

The UCL system library, which is part of the CGS delivery provides a set of predefined modules to ease SAS implementation.

Copyright per DIN 34

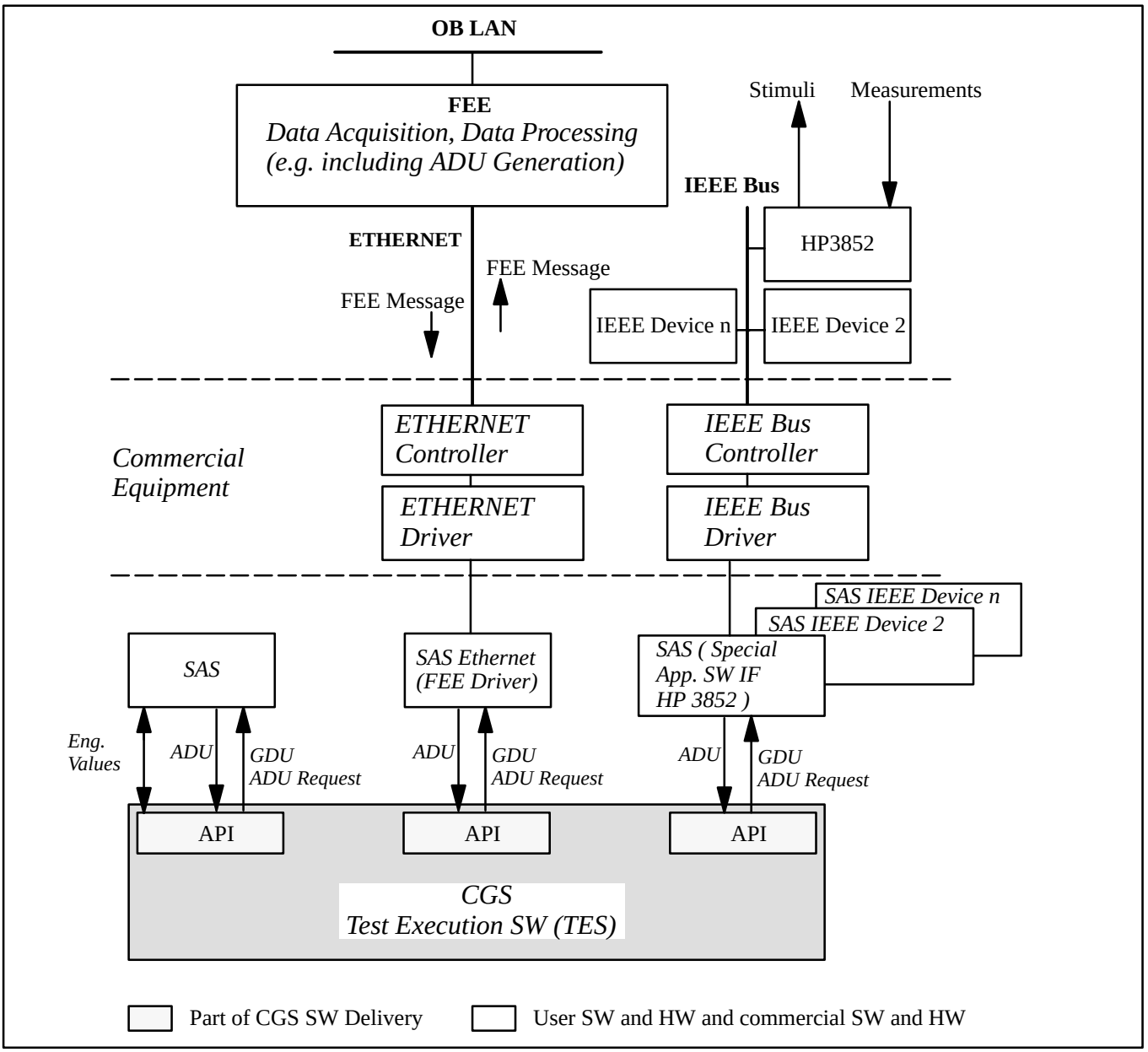


Figure 3–6 : Possible Hardware/Software Configuration for a SCOE (Test node)

3.2.2.6 Supported Modes for Test Operation Tasks and Activities

The test execution activities can be performed in three different ways:

- the **Normal** test execution mode
  in this mode the nominal tests with the UUT are performed.
  - Data is acquired from and sent to UUT
  - All input/output data is archived for replay and evaluation purposes

Simulation mode is normally used for database verification:

- the **Simulation** mode
  in this mode all incoming data is generated within the test nodes itself. The simulation of this

Copyright per DIN 34

data is driven by predefined data tables. The same functionality as in normal mode except data interface to unit under test is available:

- Input data are simulated (simulated data from unit under test)
- No output to UUT
- For verification of the configuration database and selftest
- Variable timing, i.e. same timing as in normal mode is possible
- Check automated procedures, synoptics and measurement/stimuli definitions

Note that the simulation mode described in this section may not be mixed up with the situation where a simulation model replaces the UUT.

- the **Replay** mode
  in this mode the user will see the operations of a previously executed test. The replay sub-mode shall be used to look to events anticipated by the operator which were not encounterd during tests or to investigate situations before test deviations occurred. The data presentation will be in the same way as for normal mode, but all data that are generated to interfaces external to the checkout and test system are suppressed. The replay mode may be set up with different parameters w.r.t. to timing behaviour.
  - Replay of previously recorded data i.e. real data from unit under test
  - Same functionality as in normal mode except data interface to unit under test
  - Variable timing i.e. same timing as in normal mode
  - Check automated procedures, synoptics and measurement/stimuli definitions

Replay mode can also be used for database verification.

All modes support the limit checking function, the execution of UCL commands and the visualisation of data.

Copyright per DIN 34

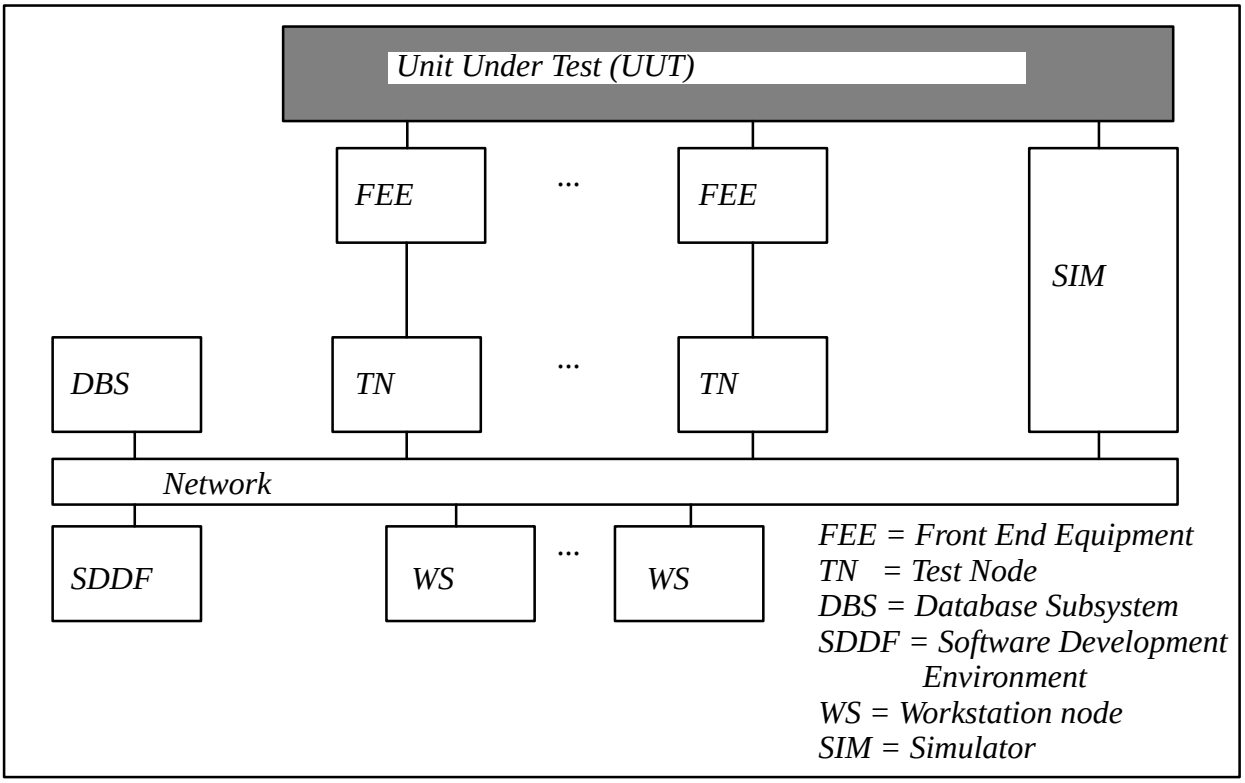


Figure 3–7 : *General Structure of a Checkout System*

3.2.2.7 Test System Control by Dedicated Languages

To support SIVQ/AIV operations, CGS Test software uses components of the COLUMBUS Language System (CLS), in particular the UCL (User Control Language) and HLCL (High Level Command Language).

Note that HLCL and UCL have the same syntax !

3.2.2.7.1 Use of the User Control Language (UCL)

UCL is defined as a set of Test object or Test system oriented commands together with a set of control commands allowing specification of automatic procedures with conditional execution. (UCL is used for both Onboard and Ground purposes).

The UCL Language

- Provides type conversions
- Supports engineering units
- Supports CGS pathname concept
- Conditional statements
  – IF, CASE, LOOP, WHILE, etc
- Libraries
  – user libraries
  – system libraries: GROUND, ONBOARD

Copyright per DIN 34

UCL allows for definition of Ground Test Automated Procedures (GTAPs) that may be activated by an interactive command and can call other GTAPs. Each GTAP is compiled offline during test preparation by the UCL Compiler. This generates an intermediate language format called i–Code. This is interpreted by the UCL Interpreter called by the Test Execution Software.

UCL supports the calling of general UCL Library routines. Such libraries may be extended by SCOE specific routines to implement SCOE dependent statements in UCL. A call to such a library routine is nevertheless defined in a definition module and imported to the UCL compiler.

The UCL GROUND System Library provides a broad range of function for following topics:

- Monitoring control
  - – change online limits
  - – enable/disable
  - – get monitor status
- Time management
  - – get local time and simulated mission time (SMT)
  - – SMT management
  - – wait
- Automated procedure control
  - – start another automated procedure (asynchronous, synchronous)
  - – suspend/resume automated procedure
  - – get status of automated procedure
  - – exchange messages between automated procedures
- SAS control
  - – start / stop a SAS
  - – change SAS mode (init, reset, etc.)
  - – get status of SAS
  - – exchange messages with SAS (synchronous, asynchronous)
- Stimulus generation
  - – send stimulus
  - – send a list of stimuli
  - – immediately
  - – ”time–tagged”
  - – enable/disable certain stimuli
- Event handling
  - – generate a log event
  - – user events
- Archive & log control
  - – enable / disable archiving
  - – close the archive
  - – enable / disable engineering value logging
- Synoptic display control
  - – show a dedicated synoptic on a dedicated screen
  - – remove a synoptic from the screen (if owner !)
- User input & output
  - – write a message to the user
  - – read a message from the user

Copyright per DIN 34

3.2.2.7.2 Use of the High Level Command Language (HLCL)

A CGS User involved in Testing operations will be able to issue interactive commands, from the Workstation, to distributed test software via appropriate User interaction methods such as windows, menus, dialog boxes etc. These commands encompass UCL statements and other Workstation or Test specific commands which together form the High Level Command Language (HLCL).

Interpreted keyboard commands issued from the High Level Command Language (HLCL) provide

- single keyboard command
- sequence of individual keyboard commands

HLCL sequences can also be defined by the User (offline) and analysed by the receiving software using the HLCL interpreter. No intermediate compilation is required. The HLCL Interpreter will access the Test Configuration Database (ie instantiation of the Mission Database) to obtain sequences of HLCL commands.

In particular, HLCL supports the invocation of GTAPs in any test node, thus establishing a further level of automation in the system as well as interactive access to each test node.

3.2.2.8 Test Evaluation Tools for On–Line or Off–Line Data Evaluation

Test Evaluation software enables the following features:

- Evaluation of data online during test operation,
- Evaluation of data generated in previous tests and for comparison of different test sessions,
- Offline resource (time, processing power, disk capacity etc) intensive evaluation functions,
- Presentation of logging data.

Test Data Evaluation (TEV) is normally performed after having executed an on–line test with the Unit Under Test (UUT) or with simulated parts of it.
The data archived often needs further evaluation in an off–line session, especially to verify wether data generated by the UUT is in the required margins or to analyse non–nominal situations during a test.

On the other hand test evaluation tools can be started to examine the data stored in the test result database during on-line test. This is possible, because data is really written to test result database ”immediately”. The only prerequisite is that the workstation on which the test evaluation shall occur must have visibility (UNIX, NFS, ORACLE) to the database. Even the same workstation that is used for test execution control can be used for test evaluation

Figure 3–8 gives an overview how data are stored and evaluated.

Copyright per DIN 34

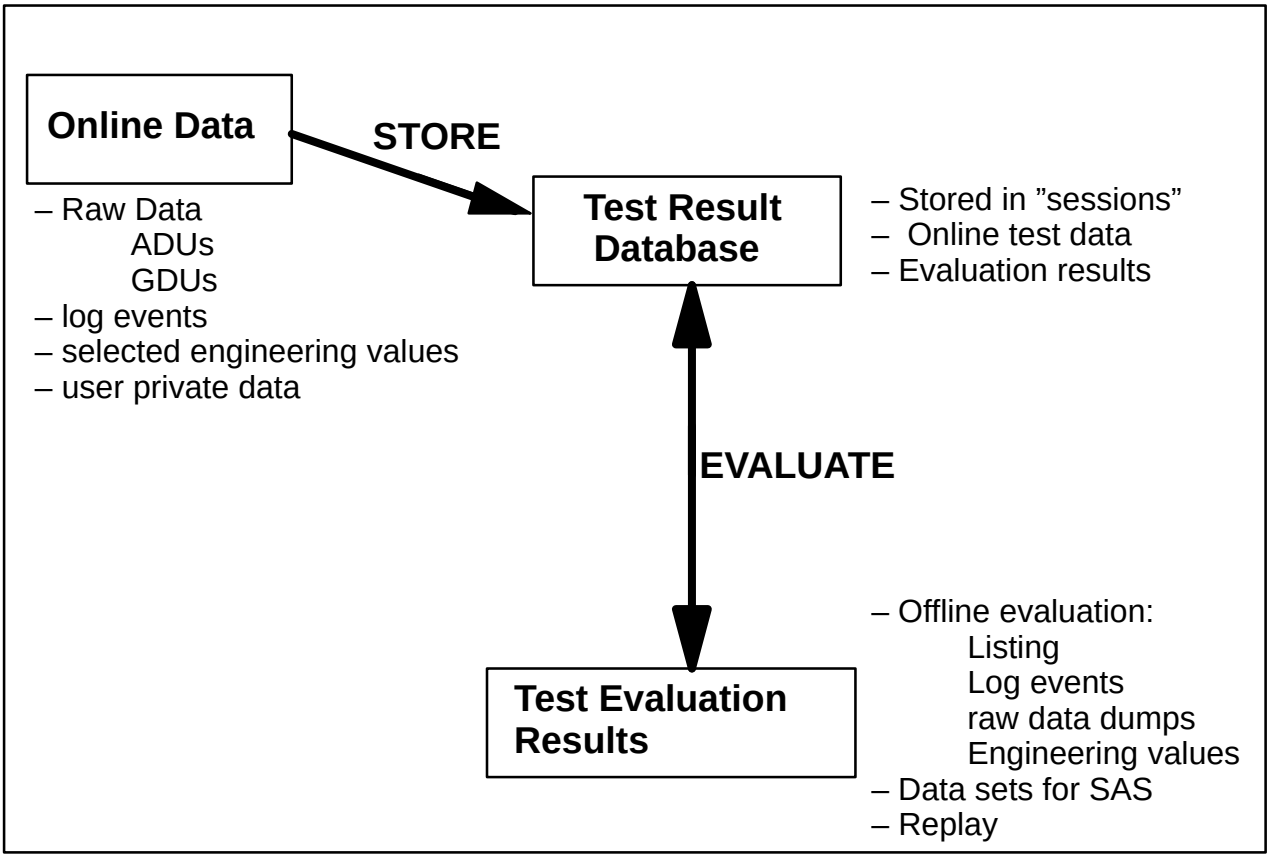


Figure 3–8 : *Data is Stored in the TRDB for Later Evaluation*

During a test following data is archived:

- raw data packets read from SAS
- data packets (stimuli) sent to SAS together with their execution status
- events needed for later replay (e.g. starting SMT)

Archive data are used for test data evaluation and in replay mode

Following data may be logged:

- error situations
- important events (e.g. sending a stimulus, system status changes)
- ”user events”
- engineering values can be logged on user request

All data is stored in the test result database.

Note that all data logically belongs to a so–called ” test session ”.

Copyright per DIN 34

3.2.2.9 Test Evaluation

In the beginning of a TEV session the user must define the data area in the Test Result Database (TRDB) to be evaluated. This might be a time frame, or simply the name of a test session, specific selection criteria and combinations of them. Additionally the user has to select a CCU version. This may be the same as used during on–line test, but it is also allowed to select an other CCU version.

The user accesses the test result database

- to obtain recorded test data (raw data, data packets, enditem engineering values, log events)

The user accesses the configuration database

- to obtain the end item's configuration data

In addition to that the user is also allowed to specify an "Evaluation Session Name". Under this name he/she can store all results of his/her work in the TRDB for later review.

TEV tools are available which allow inspection of the data and to select the data according to predefined or specified selection criteria.

TEV provides the following features in this area:

Logging Event Tool
This is the tool for detailed analysis of logged events, such as monitoring exceptions or keyboard commands. It produces listings according to selection criteria provided by the user. The output can be viewed on the screen or saved into result files.

Raw Data Dump Tool
This tool displays raw data fetched directly from the archive files. The user can specify the output format (ASCII/Hex/Dec) for a packet by packet display. The output can be viewed on the screen or saved into result files.
This tool allows the user to perform low–level debugging, e.g. to verify the communication protocol between ground and on–board system.

Data Set Tool
With this tool the user can sample signals from archived data, i.e.
– analog values
– digital values
– string values
The data will be presented with
– the raw value
– the engineering (calibrated) value
– logging time tag
– sampling information (how it was specified by the user)
For such a data collection, TEV offers additional tools for analysis:
– ***Listing Tool:*** comfortable layout generator
– ***Statistic Tool:*** derives standard statistical results upon data set parameters
– ***Graph Tool:*** drawings for data set parameters (line,bar,pie)

Copyright per DIN 34

4 CGS GENERAL TASKS

Beside the tasks that have to be performed to develop and test a system the CGS user first needs to do some basic activities to setup CGS. These activities are :

- Organise themselves
- Organise other users
- Administration of the work environment
- Error message management
- Free documentation

The following sections describe these activities in detail. The user should first carefully read section 4.1 and if necessary consult the local system administrator regarding the prerequisites prior to using CGS. These pre-requisites include a correct installation of CGS as well as the correct setup of the user environment for each user and the correct hardware– and commercial software setup and installation..

4.1 Prerequisites

See Installation Manual reference in chapter 2

- **Operating System Installation**

  Deleted: see Installation Manual

- **Desktop Installation**

  Deleted: see Installation Manual

- **CGS Installation**

  Deleted: see Installation Manual

Copyright per DIN 34

4.2 Starting CGS and CGS Task Selection

If you login to your workstation for the first time, you have to start the CGS start center by typing the command cgs in a unix shell.

> cgs

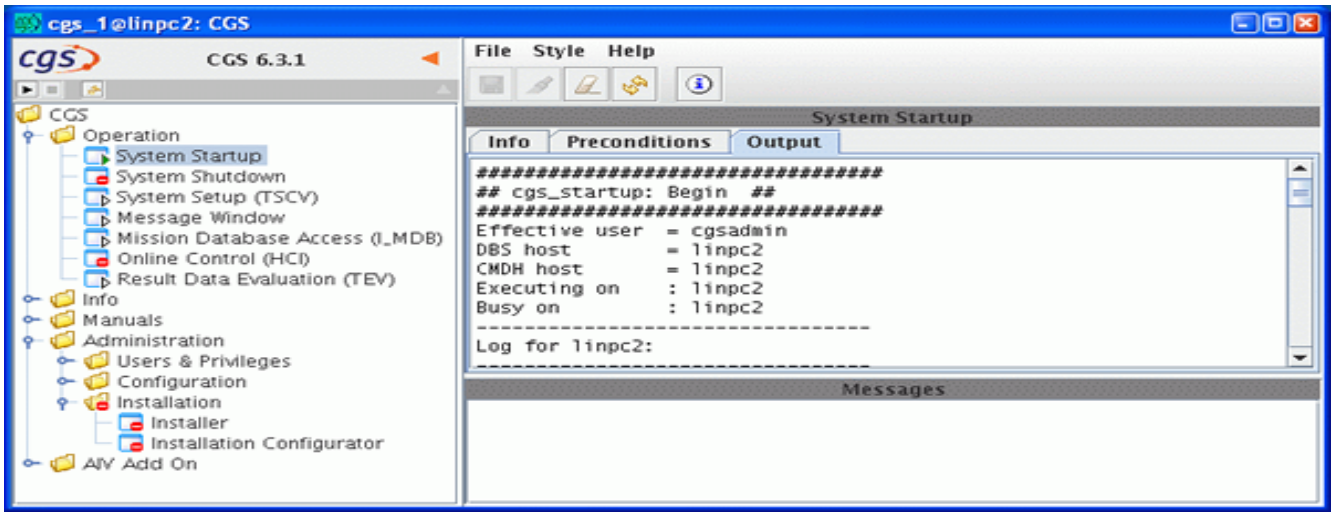


Figure 4–1 : *The CGS Start Center*

For details about the start center and configuration possibilities read the Start Center documentation in CGS–RIBRE–MA–0007. The complete CGS documentation is online available in start center in menu Manuals.

4.2.1 Message Window

The selection **Message Window** from CGS start center starts the error services application, which gives you a focus on all error messages issued from the CGS applications. You have a lot of possibilities to fine tune the reporting of error messages.

Copyright per DIN 34

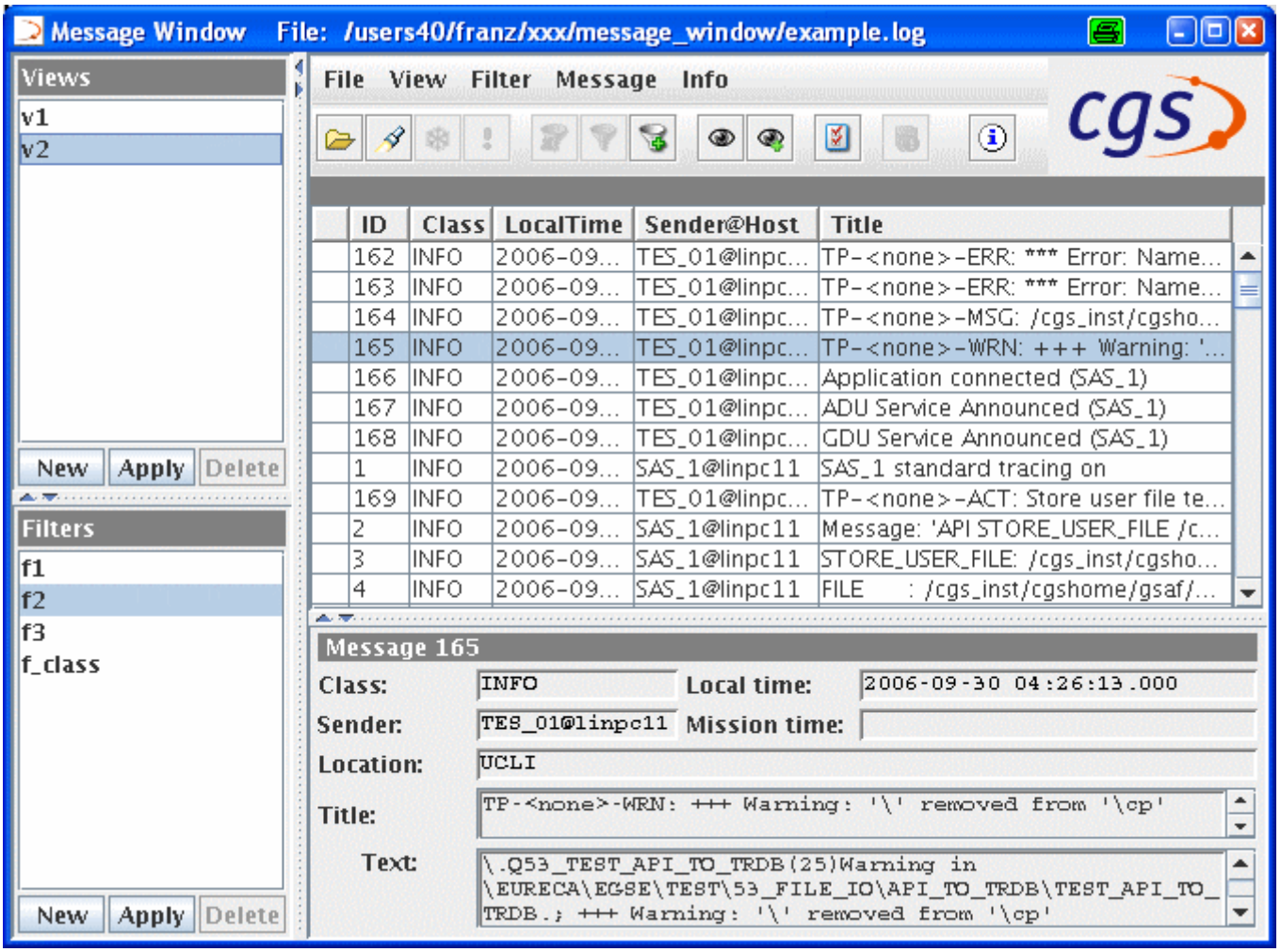

Message Window File: /users40/franz/xxx/message_window/example.log
Views
v1
v2
New
Apply
Delete
Filters
f1
f2
f3
f_class
New
Apply
Delete
File View Filter Message Info
ID Class LocalTime Sender@Host Title
162 INFO 2006-09... TES_01@linpc... TP-<none>-ERR: *** Error: Name...
163 INFO 2006-09... TES_01@linpc... TP-<none>-ERR: *** Error: Name...
164 INFO 2006-09... TES_01@linpc... TP-<none>-MSG: /cgs_inst/cgsho...
165 INFO 2006-09... TES_01@linpc... TP-<none>-WRN: +++ Warning: '...
166 INFO 2006-09... TES_01@linpc... Application connected (SAS_1)
167 INFO 2006-09... TES_01@linpc... ADU Service Announced (SAS_1)
168 INFO 2006-09... TES_01@linpc... GDU Service Announced (SAS_1)
1 INFO 2006-09... SAS_1@linpc11 SAS_1 standard tracing on
169 INFO 2006-09... TES_01@linpc... TP-<none>-ACT: Store user file te...
2 INFO 2006-09... SAS_1@linpc11 Message: 'API STORE_USER_FILE /c...
3 INFO 2006-09... SAS_1@linpc11 STORE_USER_FILE: /cgs_inst/cgsho...
4 INFO 2006-09... SAS_1@linpc11 FILE : /cgs_inst/cgshome/gsaf/...
Message 165
Class: INFO
Local time: 2006-09-30 04:26:13.000
Sender: TES_01@linpc11
Mission time:
Location: UCLI
Title: TP-<none>-WRN: +++ Warning: '\' removed from '\cp'
Text: \.Q53_TEST_API_TO_TRDB(25)Warning in \EURECA\EGSE\TEST\53_FILE_IO\API_TO_TRDB\TEST_API_TO_TRDB.; +++ Warning: '\' removed from '\cp'

Figure 4–2 : *The Message Window*

4.2.2 Overall CGS Task Flow and Task Types

The selection options as decribed in the previous chapter (NO TAG) reflects support for an overall task flow for running CGS in a test or control application. Preparation tasks have to be complemented by set–up, execute and control, and evaluation tasks.

Dependencies between tasks and required CGS processes or tools are described in later chapters. Main CGS related terms are defined in appendix B.

The following type of tasks have to be performed:

- Prepare application required data definitions and data structures to derive an overall test and control data description (see chapter 5)
- Prepare CGS interface software, such as SAS development, CIS client definition (see chapter 6)
- Prepare graphical user interface (optional) for test and control session (see chapter 6)
- Set–up test nodes, participating services and group cooperating or participating test nodes and services (see chapter 6 and 7)
- Set–up logging tools of test and control sessions (see chapter 9)

Copyright per DIN 34

- Execute and control test and control session in batch or GUI mode (see chapter 7)
- Evaluate test and control session results (see chapter 8)

Copyright per DIN 34

5 MISSION PREPARATION TASKS

This chapter describes the activities to be performed as part of the mission preparation task.

Based on the overall system requirements specification a system is structured into its elements, sub-systems and further refined components. For all the system constituents the detailed requirements and interfaces have to be described to such a level, that the real production process can be performed on the lowest level, followed by integration and test of the components.

CGS support of the mission preparation task comprises support of the following sub-tasks:

- system and sub-system identification and configuration
- identification of to be developed items (SW and HW)
- identification and definition of necessary data, state codes, etc.
- development and maintenance of SW (Ada, C and APs)
  (Note: no SWES Product in CGS V5 and later)
- and Flight Synoptic Displays
  (Note: No FWDU in CGS V5 and later)

Most CGS customers / users have developed hardware components lasting for decades and have a lot of experience with their tools. In contrast, SW development especially in Ada is rather new. Thus it was not the intention of CGS to provide tools for hardware component development; instead the purpose of CGS is to provide tools for mission preparation in terms of mission configuration, component identification and integrated SW development for at least Ada, C, UCL and synoptic displays.

Copyright per DIN 34

5.1 MDB Mission Configuration

5.1.1 Conceptual Introduction

5.1.1.1 Mission Database Structure

The purpose of the Mission Configuration is to structure the mission into well defined components in terms of system elements, element sub-systems, which again are further broken down to a level identifying items to be developed or used. This break-down is not limited and can be only a few levels, where the lowest level is associated to, for example, an automated procedure, or it can be broken down extensively to identify, for example, a particular screw.

In case of very complex systems, such as space systems, it is obvious, that only the overall mission configuration is performed by the element-contractor – as the prime contractor –, and that the further refinement is performed under responsibility of the sub-contractors involved in building a mission. However, at the end all the bits and peaces, normally already integrated into sub-components, have to be delivered to the element contractor for final integration and check-out of the element. Thus the element contractor must have the complete overview and the overall control of all peaces forming the element.

For this purpose CGS comprises a data base, the so-called **Mission Database (MDB)**, which allows you to store the complete mission configuration as a hierarchical 'tree'. At the end of the 'branches' (ie. the 'leaves'), the real data are stored. The 'leaves' in the CGS world are called **end items**, and are of a particular end item type. All branches between element definitions and end items are of type **virtual node** or **Configuration Data Unit (CDU)**, and are used for structuring purpose.

A node or end item of the mission configuration can be accessed using a so-called **long pathname**, identifying the path in the hierarchy where it can be found. In addition a unique **Short Identifier (SID)** is associated to each node, and is also used for accessing the node.

Now lets have a look on the example in Figure 5–1. The element contractor of the element APM has identified, among others, an onboard thermal control system (TCS) as one sub-system. A sub-contractor has now the task to refine the TCS. For this purpose the **system tree** is frozen by the element contractor and is now available in a certain version. The system tree reflects an element configuration break-down for a particular mission down to a specific level, e.g. down to subsystem or assembly level.

As the next step CGS allows you to create instances of a Mission Database to be distributed to the various sub-contractors. The sub-contractors will now continue with their sub-system configuration based on their MDB instance. Since the system tree is frozen, no sub-contractor will be able to modify this, but instead a sub-contractor can modify his and only his **user-tree**, identified by a **Configuration Data Unit** (CDU). Associated to a user tree is a certain range of SIDs a sub-contractor can use. This range is controlled by the Element-contractor in order to avoid ambiguity of SIDs when the various user trees are later put together.

A CDU comprises virtual nodes for structuring purposes and end items identifying the data being part of it. There are a lot of end item types, which are grouped in so–called **domains**, in order to ensure that only end items of this domain are used by the sub-contractor. For instance the sub-contractor of TCS shall not use end item types of the Electrical Ground Support Equipment (EGSE) within his user tree. Such a domain is defined, when the CDU version is created. Note, that different versions of one CDU can have different domains.

See chapter 6.6 for more details of the CGS end iten type concept and definitions.

Copyright per DIN 34

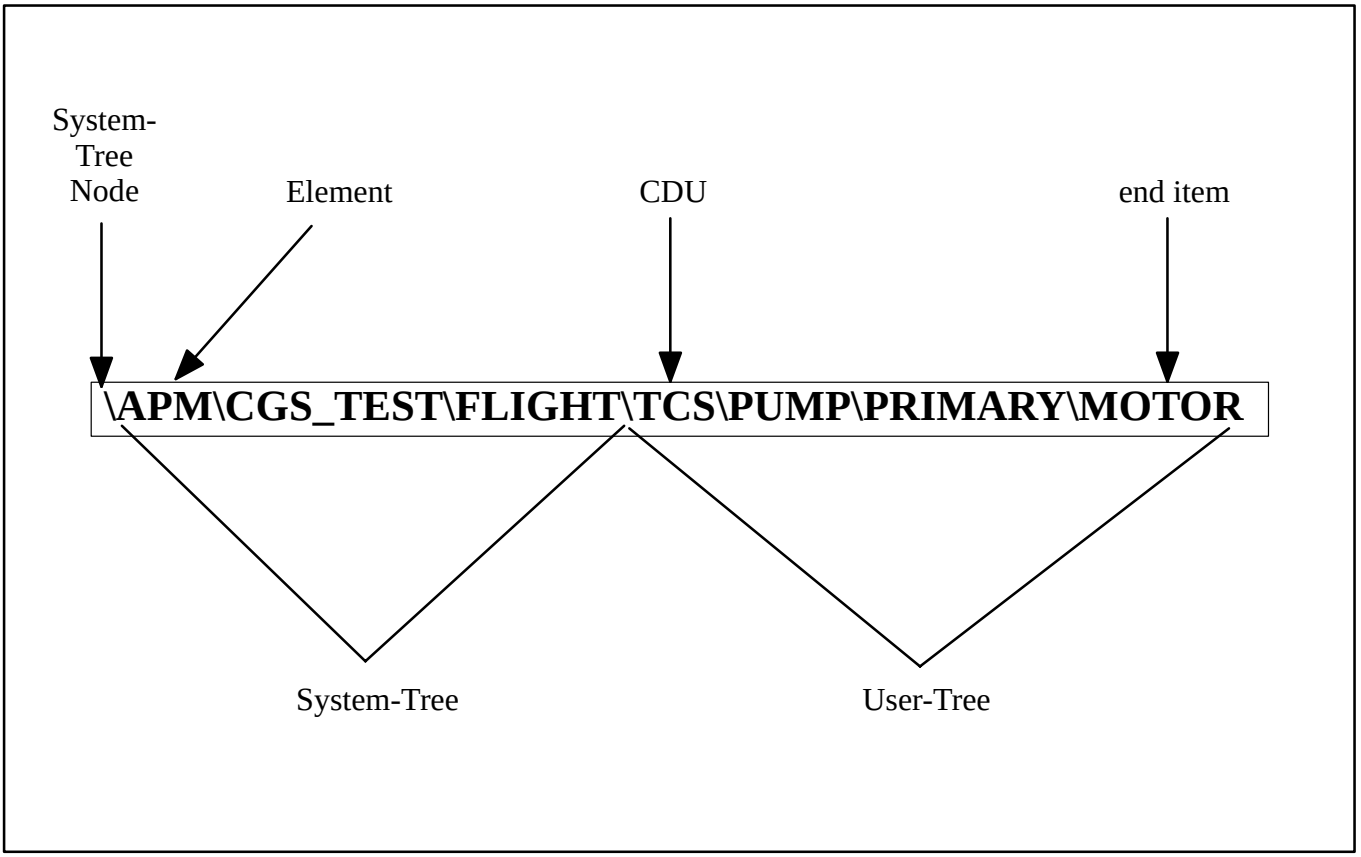


Figure 5–1 : *CGS Pathname Description*

5.1.1.2 Version Control of Mission Configurations

The term CDU version mentioned above leads to the version control concept as used by CGS MDB. This concept allows for version control on three different levels :

System Tree Version Control
During the mission preparation phase it normally happens that system requirements change for various reasons, which often changes the mission configuration as well. However, the element contractor should not just modify the existing system tree, especially if instances have already been created, rather then keeping the actual system tree persistent as an old version.

CGS supports this situation by the creation of a new system tree version, to be modified by the element contractor only. Though, if a certain system tree is referenced, it must be identified by the

System Tree Name, Mission Name, and System Tree Version

*Note, that currently the Mission Name is identified as **Dummy Mission** only.*

CDU Version Control
In contrast to the system tree modification, the CDU change is initiated by requirements, interface, and design changes as well as for bug fixing purposes. In this case it is the sub-contractor who is creating a new version, which is comprised of a leading 'V' followed by three digits separated by dots (e.g V1.2.3). Usually the version number (first digit) is increased in case of requirements changes, the issue number (second digit) in case of design or interface changes and the revision number (third digit) in case of bug fixing.

Copyright per DIN 34

CCU Version Control

Configuration Control Units (CCU) are introduced to identify a set of CDUs or even a set of CCUs, which are put together for test, simulation and/or application execution purposes. A CCU can be defined on any level between the element and defined CDUs, where CDUs can be still in the development mode. By this approach the CCU can refer to data as part of a CDU, which are even not yet defined. For this case CGS provides a consistency check facility on CCUs, which allows you to analyse the current status of a CCU.

The version of a CCU is defined the same way as for CDUs, i.e. 'V', version, issue, revision numbers. Since the definition of a CCU will not change any data, it is allowed for the element contractors as well as for subcontractors.

All of the above three version control mechanisms work only for privileged users having configuration management (cm) right. Only those users can freeze versions, whereas other users can fix a version status only by setting a version to 'for review'.

For a detailed description of the Mission Database concepts, please refer to document [2.2.1.2].

5.1.2 How to Build a Mission Configuration

This section describes how to build mission configuration by creating Elements, CDUs, virtual nodes, CCUs and end items. For detailed information please refer to [2.2.1.3]. Note, that the term 'element configuration tree' refers to both, the system tree and the user trees of a corresponding element

5.1.2.1 Starting a Mission Configuration Session

The first action to do is to invoke the appropriate CGS tool supporting the mission configuration phase. This tool is known as I_MDB (Interactive Mission Database access) and will provide the user with a window called 'I_MDB Navigator'.

The 'I_MDB Navigator' window is shown in Figure 5–2. From here you can model and navigate through element configuration trees by using the provided direct interactive access to the Mission Data Base.

Copyright per DIN 34

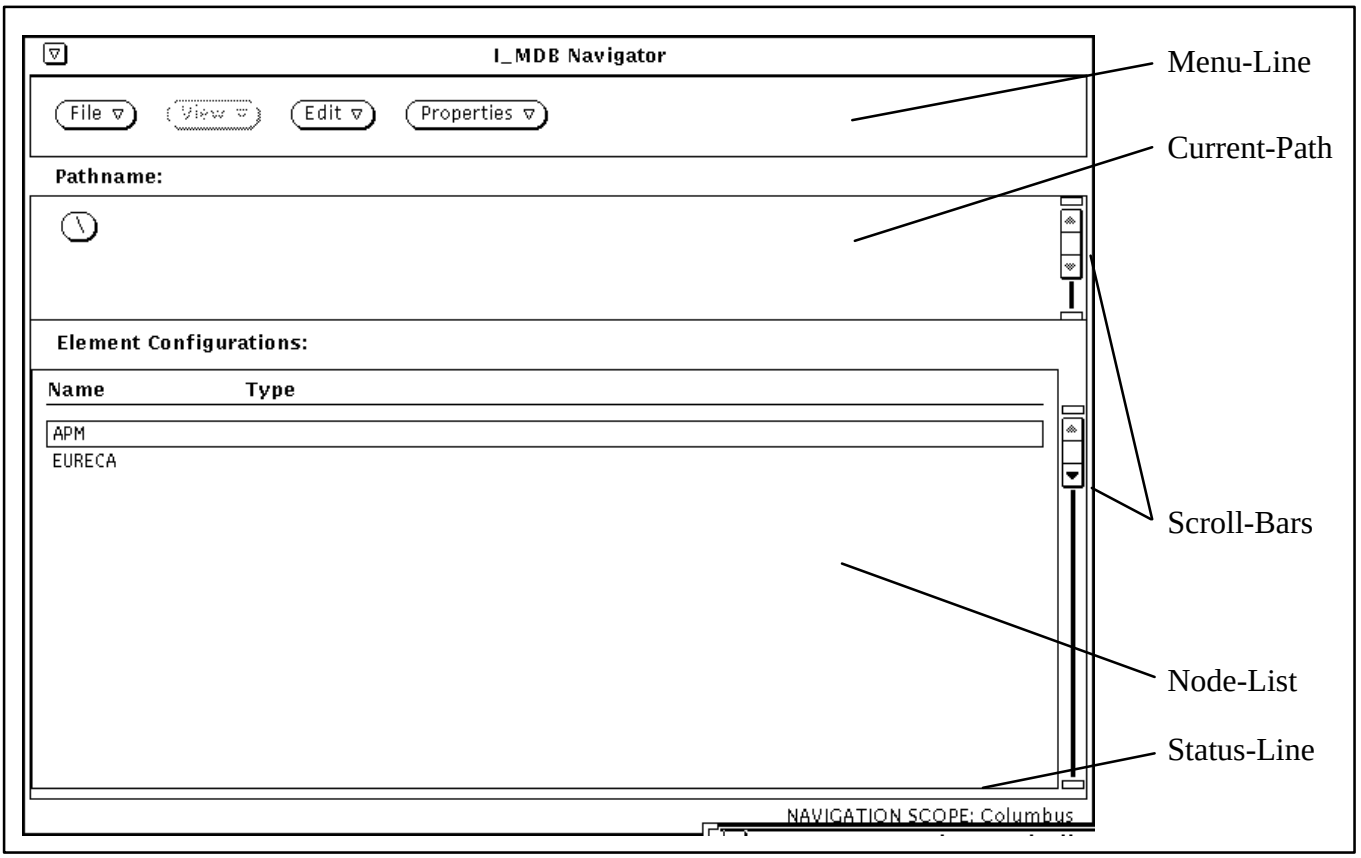


Figure 5–2 : *The I_MDB Navigator Window*

The I_MDB Navigator window is divided into several parts :

- On the top of the window the Menu-Line displays menu buttons with context sensitive functions associated. The usage is partly explained in this chapter and fully described in document [2.2.1.3].
- In the middle part, the Current-Path provides the actual path to the node the user has navigated to. After first invocation the system tree node '\' is displayed and when the user navigates through the tree, the Current-Path is updated depending on stepping down or up in the element configuration tree.
- Below the Current-Path, the Node-List provides the user with a list of currently accessible nodes being forming part of a mission configuration. Depending on the previously selected node, one of the following categories are listed :
  - all Element Configurations, if the current path is the system tree node
  - System Tree Nodes, if the current path is within a system tree of a selected element configuration
  - User Tree Nodes, if the current path is within a selected CDU
- The Status-Line provides the user with various messages.

Note, that the current version of CGS does not provide features for creation and deletion of elements. Thus you will have pre-defined elements listed under Element Configurations.

Copyright per DIN 34

The functions associated to the buttons of the Menu-Line are disabled until a selection has been made; except for the **Properties–>Tool...** button, which lists information on the tool and user. This is not explained here in detail.

The Scroll-Bars at the right site of the Current-Path and Node-List can be used in case that the provided information can not all be listed on the screen, e.g. if navigation has been performed rather deeply in the hierarchy or more than 13 nodes are listed. The usage of the Scroll-Bars is as follows:

Using Scroll-Bars in I_MDB Navigator

- **Move** the mouse to the **Scroll-Bar arrows** on the right site of the window. The arrow pointing down means scrolling down where the arrow pointing up means scrolling up.
- **Click** with the **left button** on the **arrows**. This provides you with one more line you can select from.
- **Hold** the **left button** to scroll fast up or down. This provides you with some more lines depending on how long the button is hold down.

5.1.2.2 Navigation within Element Configuration Trees

First we will explain how to navigate through the overall element configuration tree. This is necessary in order to work later with the system/user tree nodes or end items. For the following procedure it is assumed, that the I_MDB Navigator window has opened with the system tree node as Current-Path.

Navigating down within a System Tree

Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.

- **Double click** on the element you want to navigate to. This results to a box called ***System Tree Version*** listing all available version of the element system tree and the corresponding CM status.

The double click does only work on non-selected nodes (on selected nodes frame borders are visible). Otherwise the following optional step has to be performed.

Hold the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Show System Tree Versions ...*** and **release** the **mouse button**. The **System Tree Version** box pops up.

- **Double click** on the version you want to navigate to. Now the I_MDB Navigator window is updated as follows :
  - The Current-Path shows the Element Name and its system tree version.
  - The Node-List provides a list of all system tree nodes forming part of this version

if a version is already selected press the **Command button** with the right mouse button and select **Command–>Open and Dismiss Window**.

- **Double click** on the system tree node. This opens the next lower level of the selected system tree node, if this is a virtual one, or it opens a CDU (see below). The I_MDB Navigator window is updated w.r.t. Current-Path and Node-List.

or (if already selected) **hold** the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.

The above procedure works recursively for any hierarchy level within the system tree of an Element Configuration unless the selected node is of type CDU. The following procedure will now describe how to access a CDU version and the navigation within a CDU.

Copyright per DIN 34

For the procedure below it is assumed that a system tree version of an Element Configuration has be accessed and that the system tree node list contains the CDU to selected.

Navigating down within a User Tree Version

Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.

- **Double click** on the to be selected CDU. This opens a box called ***I_MDB: CDU Versions*** (see Figure 5–3) listing all the versions of the selected CDU and their status.

Also here the double click does only work on non-selected nodes. On already selected nodes (frame borders are visible) the following optional step has to be performed.

Hold the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.

- **Double click** on the **CDU version** you want to work with. This action leads back to the I_MDB Navigator window which now lists :
  - The path to the CDU version as the Current-Path
  - A list of user tree nodes forming part of the CDU version in the Node-List

or (if a CDU version is already selected) **press** the **Command** button and **select** the **Command–>Open and Dismiss Window** option.

- **Double click** on the user tree node. This opens the next lower level of the selected CDU and an updated I_MDB Navigator window w.r.t. Current-Path and Node-List.

Again, the double click does only work on non-selected nodes (on selected nodes frame borders are visible). Otherwise the following optional step has to be performed.

Hold the **right mouse button**, a menu pops-up, **move** the **mouse cursor** to the entry ***Open*** and **release** the **mouse button**.

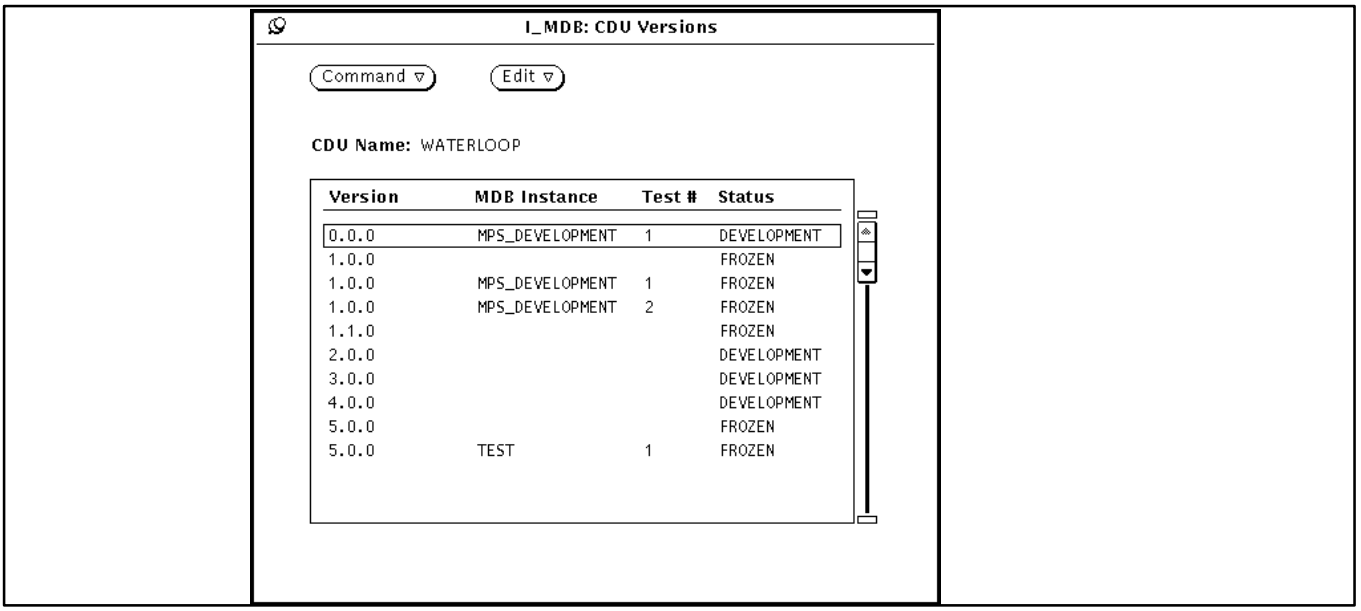


Figure 5–3 : *I_MDB: CDU Versions Box*

Copyright per DIN 34

The 'I_MDB: CDU Versions' box in Figure 5–3 lists all the versions of a CDU including CDU test versions. CDU test versions are created in case of changes shall be performed on the user tree by organisation who do not have access to the appropriate MDB instance. If a test CDU is created, this does not belong to the original MDB instance. Refer to document [2.2.1.2] for more detailed information.

A special way for navigation represents the selection of a Configuration Control Unit (CCU) within the system tree, but beneath an Element. By selecting a CCU version a special view on the element configuration tree is created, providing only those CCUs and CDUs defined for the selected CCU version.

A CCU can be selected either from a complete list of all CCU versions defined for the selected element or from a list of all CCU versions defined for a particular System Tree node :

Note that a virtual System Tree Node must be selected in the Current_path field.

Navigating down to a CCU version from the complete list

- **Move the mouse** to the Menu-Line and **select File–>Browse All CCU Versions...** . This opens the ***Browse CCU Versions*** box listing all defined CCUs including pathname, CCU Name and CCU version.
- **Use the Scroll-Bar** on the right site of ***Browse CCU Versions***. (works like for the I_MDB Navigator window, see above)
- **Click** on the CCU you want to access and **click** on the ***Apply*** button.

The I_MDB Navigator window is now updated in two parts :

- The Current-Path shows now the path nodes down to the CCU, where at the end of the path the name and the version of the CCU is displayed.
- The Node-List does now contain only
  - the CDUs
  - and the virtual system tree nodes reflecting CCUs

  forming part of the selected CCU version.

The above described update of the I_MDB Navigator window does also apply for the second way of navigating to a particular CCU version.

Navigating to a CCU of a particular System Tree Node

- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select *File–>Show CCU Versions...*** . This opens the ***I_MDB: CCU Versions*** box listing all defined CCUs which root is the selected system tree node (see example in Figure 5–4) .

  The ***I_MDB: CCU Versions*** box is divided into two parts. The left part lists the names of all CCU names being defined. By selecting one CCU, the right part is updated, listing all versions and their status created for the selected CCU.
- **Click** on a CCU in ***CCU Names*** . The right part of the ***I_MDB: CCU Versions*** box is updated with version informations.
- **Double click** on a CCU version.

Copyright per DIN 34

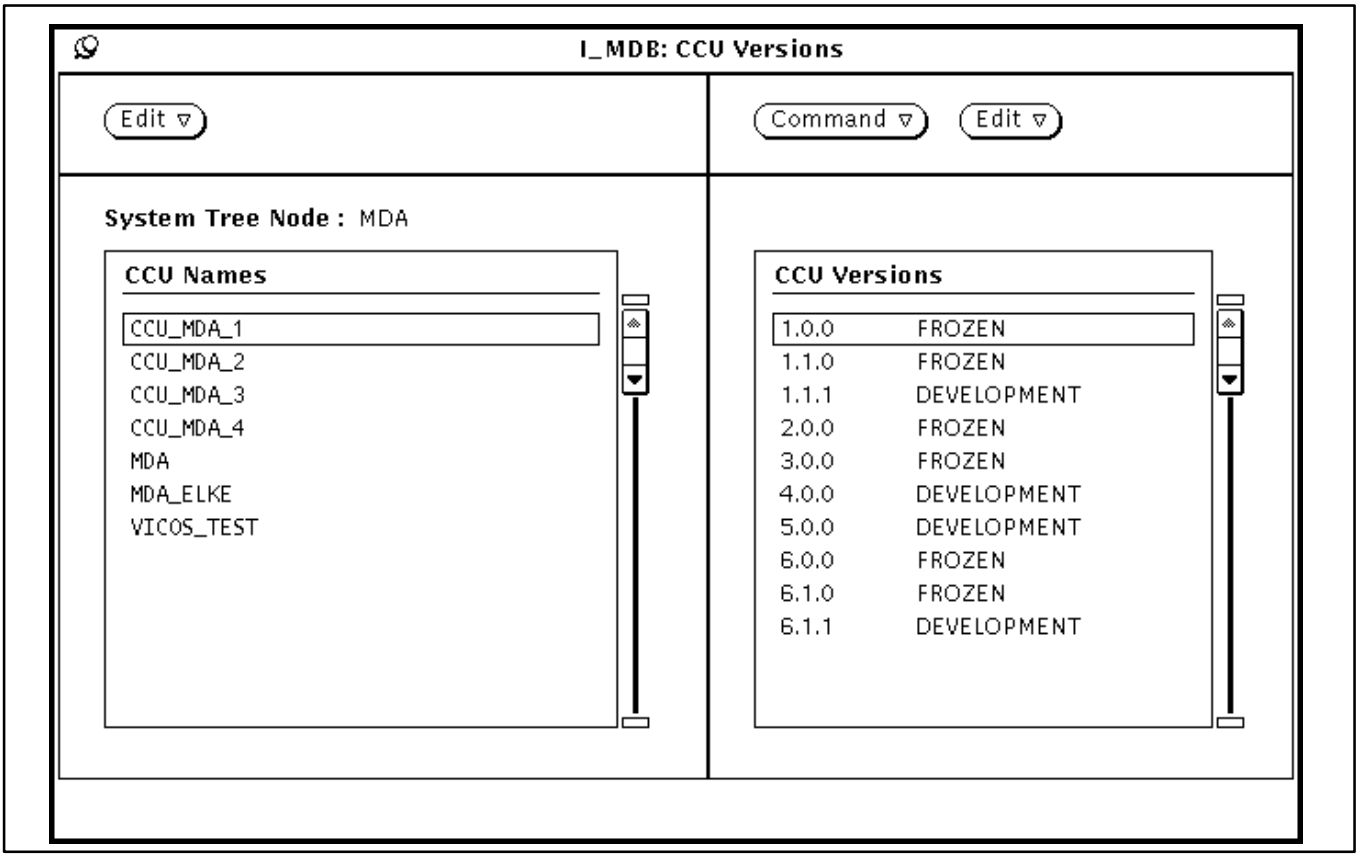


Figure 5–4 : *I_MDB: CCU Versions*

Note, that once a CCU has been selected you will not be able to select another CCU version within the current one. For this purpose you have to navigate upwards in the element configuration tree (see below)

Up to here it was described how to move down in the element configuration tree. Moving upwards is much easier and works always the same.

Navigating upwards in the Element Configuration Tree

- **Move** the mouse into the Current-Path part of the I_MDB Navigator window.
- **Double click** on the path-node you want to move to. This updates the I_MDB Navigator window with the new node informations associated to the selected path-node.

Note, if a path-node higher than a selected CCU is selected, the current CCU version is no longer applicable.

Copyright per DIN 34

5.1.2.3 Creating Nodes in a Element Configuration Tree

Since the element configuration tree as defined above being a system tree plus the corresponding user trees, the creation of nodes is foreseen in terms of

- Creation of Elements
- Creation of System Tree Nodes
- Creation of User Trees (CDU)
- Creation of User Tree Nodes
- Definition of Configuration Control Units (CCU)

However, the current version has not reached its final implementation, and therefore the following note has to be given.

The current CGS Version will provide predefined System Tree Versions which are FROZEN so that they cannot be updated or deleted in any way. Thus it is not possible to create Elements or virtual System Tree Nodes.

The creation and modification of nodes within the element configuration tree shall not be performed by any user, but only by personal with MDB CM access. This has to be checked first.

Checking CM Status

- **Move** the mouse to the Menu-Line.
- **Select Properties–>Tool**. This provides you with the tool and user properties.
- **Check** that the **CM Status** is set to **Yes**. If this is not the case, you will not be able to perform the actions described in the following.
- **Close** the ***Tool Properties*** box by **clicking** on the **pin** in the upper left corner.

The first node to be created is a CDU, which than contains user tree nodes.

Creating initial Configuration Data Unit (CDU)

- **Navigate** to the system tree node where a new CDU shall be placed.
- **Move** the mouse to the Menu-Line and **select File–>Create CDU...**. This will provide you with the ***Create CDU*** box.
- **Enter** the <*cdu–name*> and **click** on ***Apply.*** Now the new CDU is placed in the list of System Tree Nodes.

Now with the above procedure only an initial CDU has been created, but a CDU version has not been defined yet. This is done by the following procedure which works also for CDU where versions are already defined.

Copyright per DIN 34

Creating a new CDU Version

- **Double click** on the newly created CDU. This opens an empty ***I_MDB:CDU Versions*** box similar to Figure 5–3 but without any contents.
- **Select** ***Edit–>Create...*** from the menu buttons on the top of the box. This opens the ***Create CDU Version*** box. A detailed description is provided below.
- **Click** on one of the 4 options to specify a new version in the ***Create CDU Version*** box. The selection will be highlighted by a bold frame.
- If the new version shall be a copy from an old one, click on the ***Copy from CDU Versions...*** button. This provides you with the ***I_MDB: CDU Version list*** box. This looks similar to Figure 5–3, except that there is an ***Apply*** button at the bottom.
- **Click** on a version listed in the ***I_MDB: CDU Version list*** box and **click** on ***Apply.*** The version number, incl. the test CDU version reference (if applicable) is displayed behind the ***Copy from CDU Versions...*** button.
- **Click** on the **CDU domain...** button to define the domain. This provides you with the ***CDU domain list help*** box, listing all possible CDU domains.
- In the ***CDU domain Help list*** box, **click** on the CDU domain to be selected.
- **Click** on the ***Apply*** button. Now the name of the CDU domain is displayed behind the ***CDU domain...*** button in the ***Create CDU Version*** box.
- *The definition of a CDU domain must not be performed in case of copy a CDU version.*
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner.
- **Click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create CDU Version*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create CDU Version*** box. Now the latter box disappears and the new CDU version is listed as part of the ***I_MDB: CDU Version*** box.

The ***Create CDU Version*** box (Figure 5–5) contains several items:

- In the first row you have to decide if you want to create a new version, issue, revision or a new test version. If a new CDU is created, a new version must be created in any case !.
  If a new issue or revision shall be created for an existing version, the latter must have the status **FROZEN**.
- The ***CDU domain*** is a mandatory field, which definition defines the set of end items which can be created within this CDU version. The domain is decided automatically in case that a copy from CDU versions has been performed.
- The ***Copy from CDU Versions*** must be performed in case that a new issue or revision shall be created. In other cases the can be filled optionally.
- The ***Owner*** is also a mandatory field to decide the owner of a CDU version. The owners of different versions of a CDU may differ.
- The field ***Description*** can optionally be used for comments to a CDU version.

Copyright per DIN 34

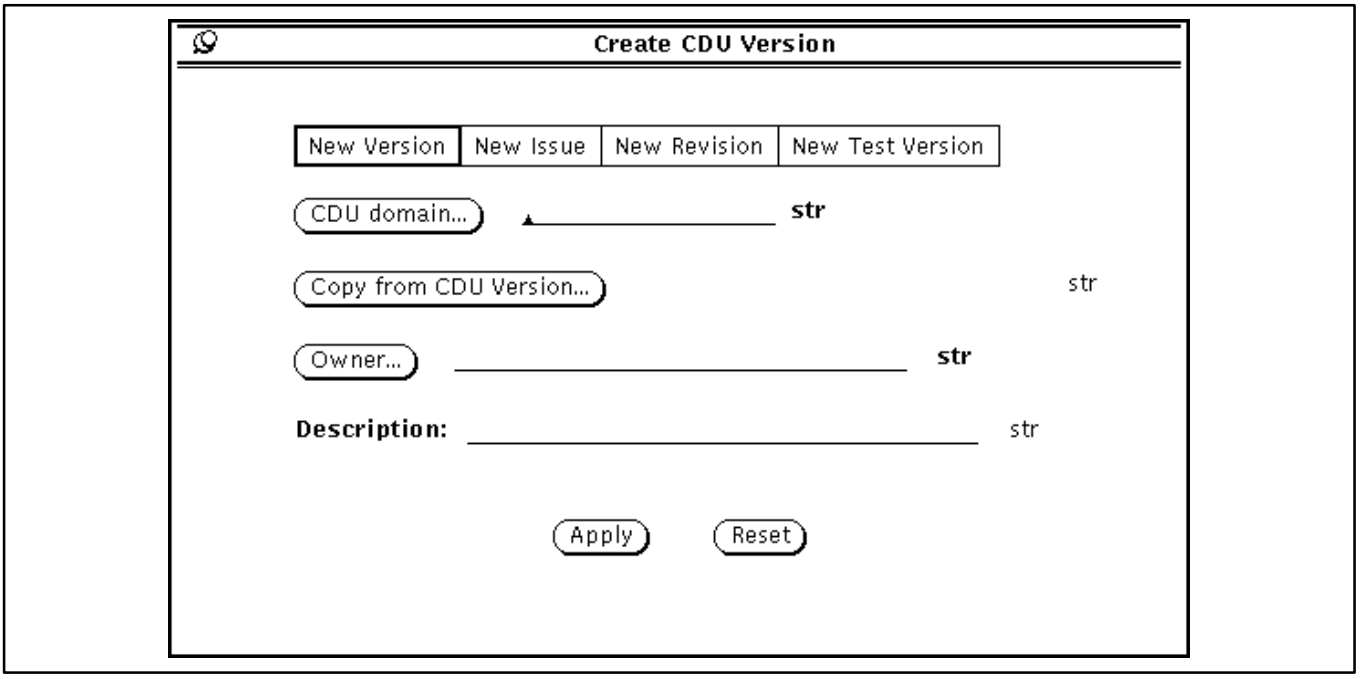


Figure 5–5 : *Create CDU Version Box*

Having accessed a CDU version, new User Tree Nodes can be created within this user tree. User Tree Nodes are either virtual or end items. The user tree node type 'virtual' is the default type. End item types have to be selected from a list of available end items depending on the CDU domain. As part of this section only the creation of the virtual user tree nodes are described, because those are used to structure the overall mission configuration. The creation of End Items is explained in section 5.2.

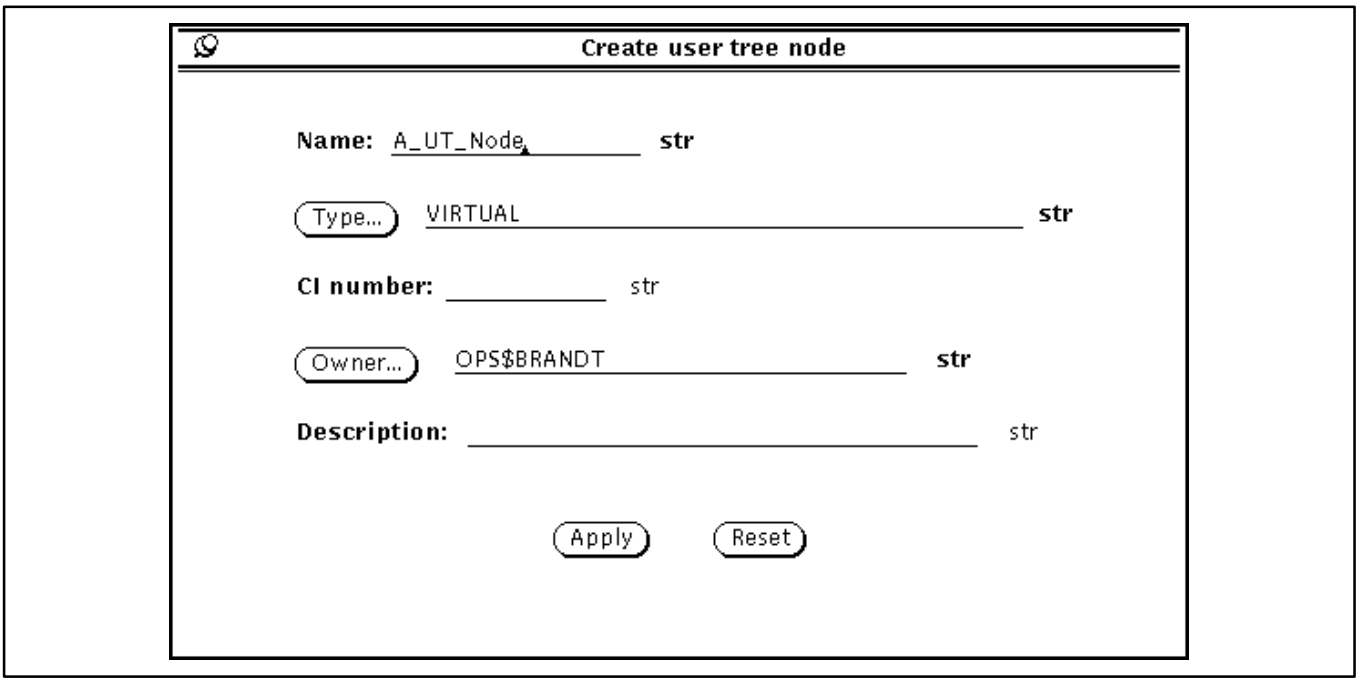


Figure 5–6 : *Create User Tree Node Box*

Copyright per DIN 34

Creating User Trees Nodes

- **Navigate** into a particular CDU version.
- **Select File–>Create Node...** in the command part of the I_MDB Navigator window. This opens the ***Create user tree node*** box (see Figure 5–6).
- **Enter** the <*user–tree–node–name*> behind the entry **Name.**
- **Click** on the ***Type...*** button. This will provide the ***Node type list help*** box, listing all possible node types.
- In the ***Node type list help*** box, **click** on the desired node type, then **click** on the ***Apply*** button. Now the node user nametype is displayed behind the ***Type...*** button in the ***Create user tree node*** box.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner, then **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the new user tree node is listed as part of the Node-List of the I_MDB Navigator window

All the creation of CDUs and user tree nodes is mainly done to structure the lower level mission configuration. For the purpose of application execution and test it is necessary to select a set of CDUs and even CCUs versions to create a new CCU version.

Create a new Configuration Control Unit (CCU)

- *Note that virtual tree nodes can be the root of a CCU only.*
- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 5–4) listing all defined CCUs which root is the selected system tree node.
- **Select Edit–>Create...** on the upper left site of the ***I_MDB: CCU Versions*** box. This provides the ***Create CCU*** box.
- **Enter** the <*new–ccu–name*> in the ***Create CCU*** box and **click** on the ***Apply*** button. Now the new CCU is listed on the left part of the ***I_MDB: CCU Versions*** box.

Copyright per DIN 34

Create a CCU Version

- *Note that virtual tree nodes can be the root of a CCU only.*
- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 5–4) listing all defined CCUs which root is the selected system tree node.
- **Click** on one of the listed CCU names at the left part of the **I_MDB: CCU Versions** box. This will list all available CCU versions of the selected CCU on the right side.
- **Select Edit–>Create...** on the upper right site of the ***I_MDB: CCU Versions*** box. This provides the ***Create CCU Version*** box (see Figure 5–7).
- **Click** on the **New Version** option to specify a new version in the ***Create CCU Version*** box. The selection will be highlighted by a bold frame.
- *To create a new issue or a new revision **click** in the preceding step either the button **New Issue** or the button **New Revision**.*
- **Click** on the ***Copy from CCU Versions*** button (this is optionally if a new version shall be created). This opens the ***CCU Version list*** box listing all available CCU versions for the selected .
- **Click** on a CCU version in the ***CCU Version list*** box and **click** the ***Apply*** button. This copied version number is displayed behind the ***Copy from CCU Versions*** button.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create CCU Version*** box.
- **Enter** some <*description test*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create CCU Version*** box. Now the latter box disappears and the new CCU Version is listed as part of the right site on the ***I_MDB: CCU Versions*** box.

Copyright per DIN 34

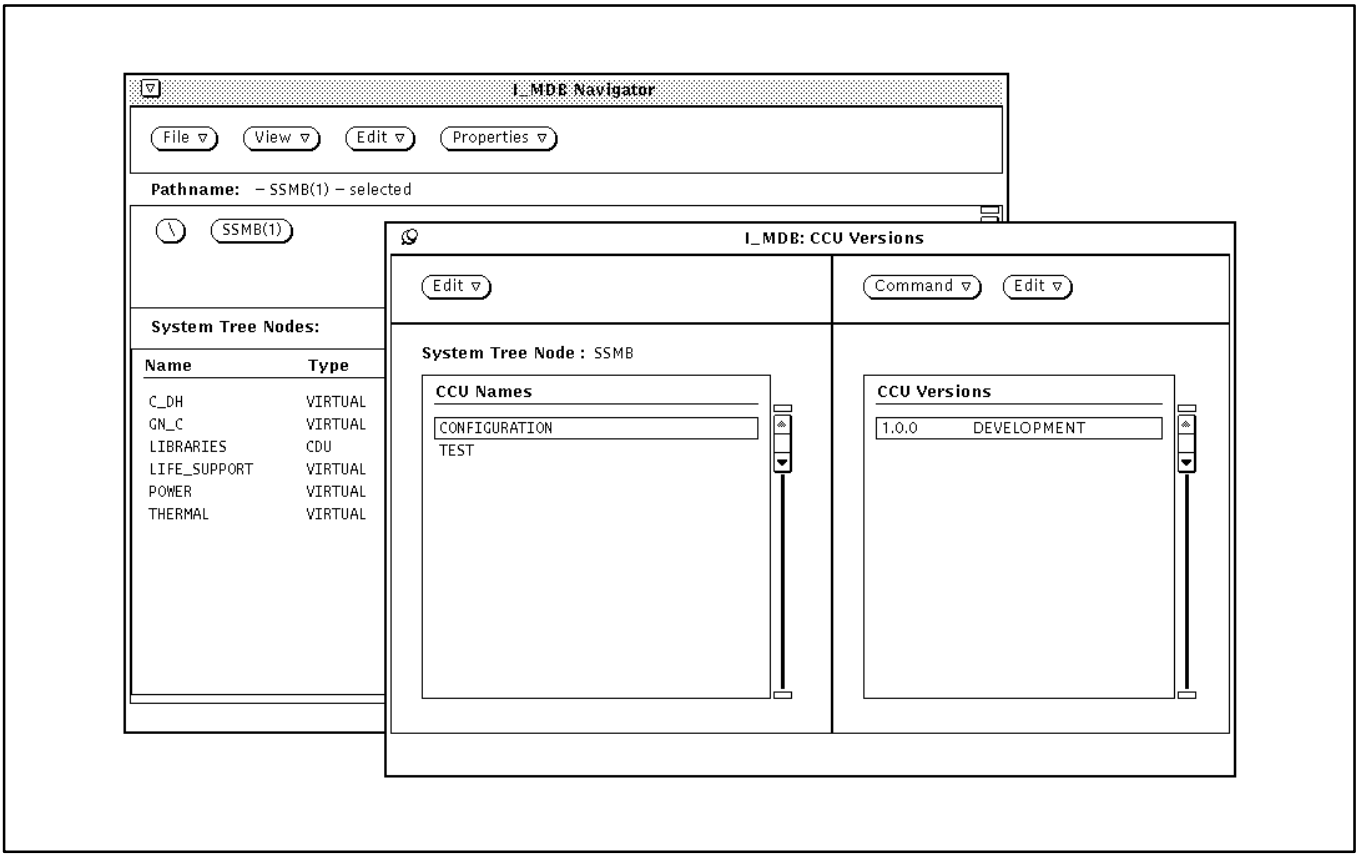


Figure 5–7 : *Create CCU Version Box*

Now a new version of a CCU is created, but the contents is not necessarily defined. This is especially true if a completely new CCU has been created or the CCU version has not been copied from an existing ones, because then the CCU version is empty. In the following it is described how the contents of the CCU is specified.

Copyright per DIN 34

Specifying CCU Version Contents

- **Navigate** to the path-node which shall be the root of a CCU.
- **Click** on the last entry of the Current-Path (must be a virtual system tree node)
  or
  click on a virtual system tree node within the Node-List of the I_MDB Navigator window.
- **Move the mouse** to the Menu-Line and **select File–>Show CCU Versions...** . This opens the ***I_MDB: CCU Versions*** box (see Figure 5–4) listing all defined CCUs which root is the selected system tree node.
- **Click** on one of the listed CCU names at the left part of the ***I_MDB: CCU Versions*** box. This will list all available CCU versions of the selected CCU on the right site.
- **Click** on one of the listed CCU versions at the right part of the ***I_MDB: CCU Versions*** box and **select Command–>Maintain References...** from the menu buttons on top of the box. This opens the ***I_MDB Maintain References*** box.

  The ***I_MDB Maintain References*** box contains two parts. The upper part lists all referenced CCU versions, where on the bottom all CDU references are listed. The following steps are exactly the same for the creation of both kind of references.
- **Click** on the ***Edit*** button for CCUs/CDUs. By this the ***Browse CCU/CDU Version*** is displayed, listing all CDU/CCU created beneath this system tree node.
- **Click** on a CCU/CDU you want to refer and **click** on the ***Apply*** button. Now the referenced CDU/CCU is listed in the appropriate part of the ***I_MDB Maintain References*** box.
- When all references have been established, **click** on the **Pin** on the upper left corner of the ***I_MDB Maintain References*** box
- **Select** the **Command–>Open and Dismiss Window** to navigate to the maintained CCU version.

*The selected CCU is displayed in **bold** type in the Current_path part of the I_MDB navigator.*

This finalizes the section on how to create user trees, user tree nodes and CCUs. For more detailed information please refer to the corresponding detailed user manual [2.2.1.3].

Copyright per DIN 34

5.2 MDB Creation and Contents Definition of Data

5.2.1 Introduction

The definition of data is performed on an Element Configuration by creating end items as part of the user trees (CDU), that are also stored in the Mission Database (MDB) . End items represent the lowest level individual reconfigurable items like automated procedures or commands and are modelled by the leaf nodes of the user tree.

In contrast to virtual items (refer to section 5.1.1), end items contain all detail information describing specific end item characteristics. Thereby end items can be comprised to different end item classes (end item types) where all end items of a class (type) are characterized by the same set of attributes.

Example1:
All automated procedures are described by the attributes parameter list, source code, etc.. A specific automated procedure 'Activate Pump of the APM TCS' will be stored in the MDB together with its concrete parameter list, source code, etc..

Example2:
If the same error message information is used for several limit sets, it is unsatisfying for a user to be forced to enter exactly the same error message data again and again for each of these limit sets. In this case it is rather desirable that the user once defines an end item of an MDB provided end item type 'TCS Error Message' which can then be referenced by all corresponding limit sets.

Especially Example 2 shows that the constitution of end item classes and the ability to reference end items are extremely useful in avoiding redundancies.

A lot of end item classes are predefined in the MDB together with the attributes which characterize each class. In addition CGS provides for the possibility to create new end item classes, but it is important that those should base on the existing classes. This is especially important if those end items shall later be used within an Engineering Ground Support Equipment (EGSE) configuration of CGS to be used for final check-out.

- *Note, if a new end item class does not base on an existing one, a small utility has to be developed by at CGS customer/user site to map the new end item class to existing end item classes.*

For this overall CGS User Manual the customization of end items is not described further. Please refer to document [2.2.1.3] for more information on this subject.

From the CGS users point of view the definition of end items is done in different procedural behaviour, which can be grouped as follows :

1. End items containing purely a set of data. For those end items various masks are provided where the user can enter the relevant data. This is for most of the end items.
2. End items invoking special tools necessary to create the relevant data. These end items are those for the definition of automated procedures, simulation models and synoptic displays.
3. End items referring to Software Replaceable or Software Exchangeable Units (SWRU, SWEU). In this case parts of a data set are down-loaded from the SDE repository into the MDB.

In the following the various procedures are described in separate sub-sections.

- ***We should like to put your attention to the appendix 6.6.4, where most end items are described.***

Copyright per DIN 34

5.2.2 How to Define End Items Containing Data

As already mentioned above data definitions are represented as end items in the MDB, which are placed beneath a specific user tree, i.e. in a CDU version. Thus before an end item shall be created you have to navigate to the path-node of a user tree you are allowed to work with. Here you can now create a new node and select the specific end item type (different of type 'virtual').

Caused by the fact that their are quite a lot end item types and because the user can define their own end item types, the creation of an end item is only performed exemplary, i.e. we will create an end item of type General Purpose.

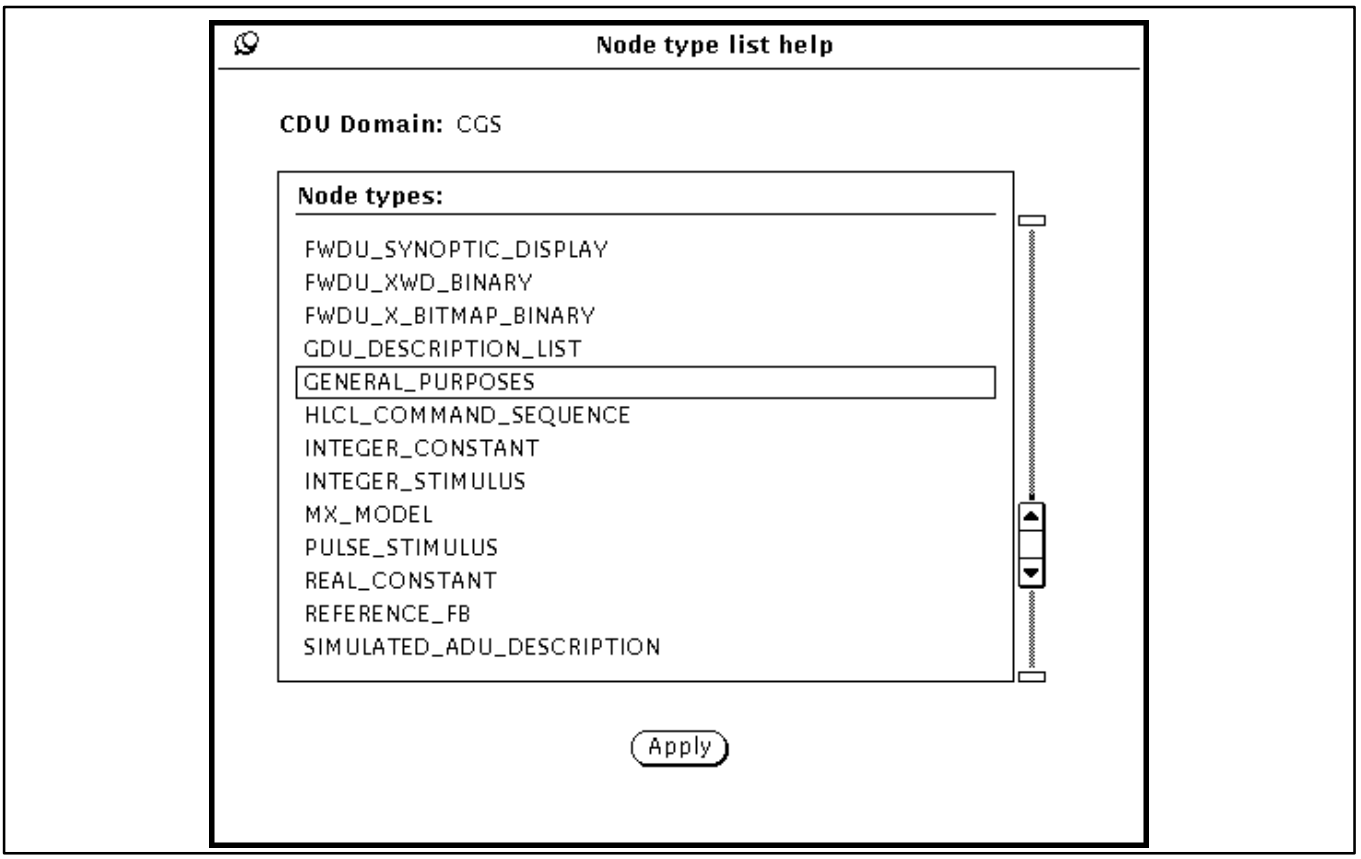


Figure 5–8 : *Node Type List Help*

Creating an End Item (exemplary of type General Purpose)

- **Navigate** to a user tree node as described in section 5.1.2.2.
- **Select File–>Create Node...** in the command part of the I_MDB Navigator window. This opens the ***Create user tree node*** box (see Figure 5–6).
- **Enter** the <*name*> behind the entry **Name.**
- **Click** on ***Type...***. button. The ***Node type list help*** box listing all end item types associated to the CDU domain (see Figure 5–8) appears. For this procedure the type GENERAL_PURPOSES is selected.

continued ...

Copyright per DIN 34

- **Click** on the a.m. end item type and click on the ***Apply*** button in the ***Node type list help*** box. The node type 'virtual' in the ***Create user tree node*** box is now replaced by GENERAL_PURPOSES.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description text*> behind the field ***Description :***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the AP is listed as part of the Node-List of the I_MDB Navigator window

SInce now the new end item is identified, the next step is to define the contents of the end item. The creation of data and thus the available operations are strongly dependent on the end item type. However, there are still some general operations available, as explained in Table 5–1, which apply for all end items and even for virtual nodes.

Operation	Description
Open	In case several data are to be specified directly on the end item, the Open operation provides further sub-menues. In case of a virtual node, Open navigates to the next lower level.
Pathname & SID...	Show the end item Short Identifier (SID) and the corresponding pathname, but can also be used to view the pathname of a specified SID.
Tools	Depending on the End Item Type an appropriate tool (program) is invoked. Using this tool the data of an end item are created / modified. (see also the following sections)
Delete	Removes the node /end item from the current CDU version.
Copy	Copies the node /end item. The copy functions works also on sub-trees in case that a virtual node is further refined.
Paste	Pastes a copied node / end item into the current CDU version.

Table 5–1 : List of General Operations on Nodes including End Items

The access of the operations is quite easy and shown in the following procedure. In this case it is a general description not based on the above example, except, that the exemplary shown display of end item operations (see Figure 5–9) is based on it.

Not described inhere, are the several tools and data entry boxes, because, again, these are depending on the end item types and are either described in other parts of this manual or can be derived from the data description in chapter 6.4.2

Copyright per DIN 34

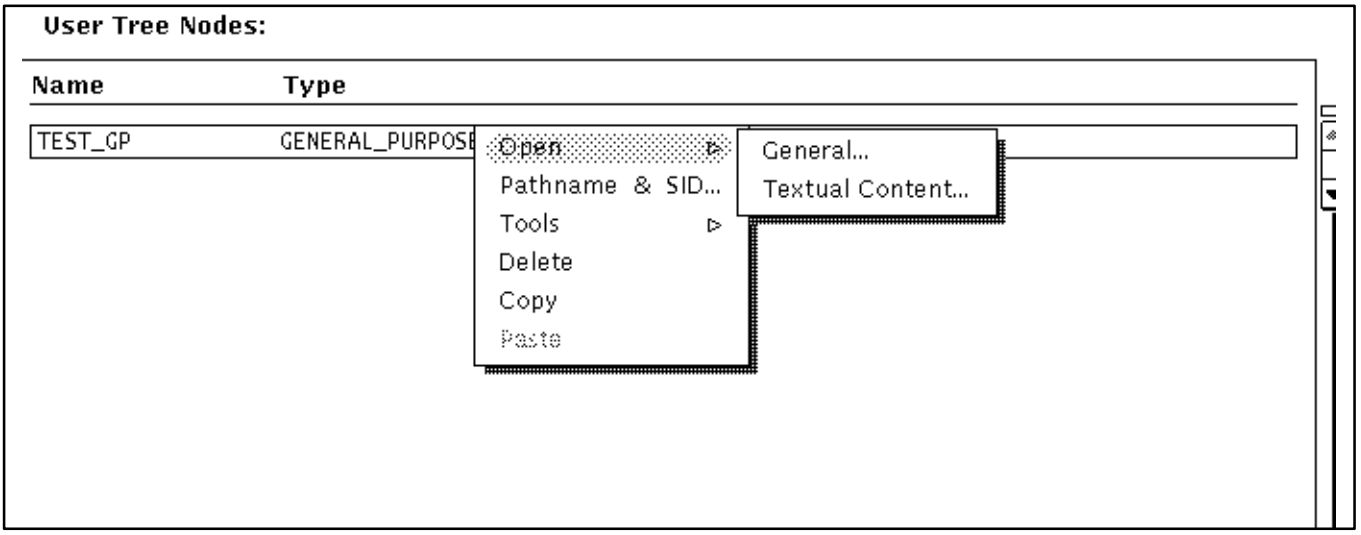


Figure 5–9 : *Exemplary Shown End Item Operations with the I_MDB Navigator Window*

Accessing Operations on End Items.

- **Navigate** to the node you want to access.
- **Click** on the to be accessed node of the Node-List in the ***I_MDB Navigator*** window.
- **Hold** the **right mouse button**. A ***menu box*** appears on the screen.
- **Move** the **mouse cursor** to the small **arrow** behind the ***Open*** operation. This will provide you with a ***sub-menu box***.
- **Select** an operation of the ***sub-menu box***. Here the operations are different for each End Item type.
- **Release** the mouse button. This will provide you with the corresponding data entry box.
- In case another window is opened which does not allow for direct data entry, **use** the ***menu buttons*** in an appropriate way.
- **Enter** the data to the corresponding fields as described in appendix 6.6.4.
- *Note, if End Item data entry masks provide data of type STRING, be careful with copy / cut and paste of text, because if text is copied longer than the allowed size, the CGS tool (MDA) will exit with an oracle error message.*

Copyright per DIN 34

5.2.3 How to Maintain the End Items Data (Interactive)

To enter data into the enditems' aggregates, the I_MDB tool provides access to each aggregate via the "Open" menu entry.

In addtion, the Detailed Data Editor (DDED) may be called which opens all aggregates for an enditem and allows data manipulation by grouping related aggregates and showing aggregates in parallel windows.

The DDED is to be called via the "Tools" menu entry for each enditem.

5.2.4 How to Maintain the End Items Data via Batch Operation

Enditem may be created and loaded also from ASCII files using the Batch Data Entry Tool of CGS. BDE allows to fill each aggregate of an enditem via data files generated outside of CGS or in other CGS environments, calling the Batch Data Output Generation (BDO), which creates the enditem's data into ASCII files directly suitable for BDE input processing on other CGS sites

Copyright per DIN 34

5.2.5 Defining CLS Related End Items

5.2.5.1 Introduction

For definition of items of several end item types a tool named ”CLS Editor” is used. It processes items of type UCL_AUTOMATED_PROCEDURE, UCL_USER_LIBRARY, UCL_SYSTEM_LIBRARY, HLCL_COMMAND_SEQUENCE, CPL_SCRIPT, EGSE_xxx_DERIVED_VALUES and items which have parameter lists but no execution source code.

Automated Procedures (APs) are software programs written in the User Control Language (UCL) as defined in the UCL Reference Manual [2.4.3]. UCL acts as a pure programming language: automated procedures and libraries are edited off–line, compiled and kept in the database for later on–line execution.
With such programs command sequences used on the flight system or test procedures used in the check-out phase can be automated.

Thus it provides also a library concept, whereby there are fixed **UCL System Libraries**, providing general functions and procedures in UCL and the possibility of **UCL User Libraries**, which can be created and used by end-users of CGS. In the UCL User Libraries the end users can define their own functions and procedures as used by various automated procedures.

The formal parameter List definition for an item must also be written in UCL as defined in the UCL Reference Manual [2.4.3].

The expressions for derived values must also be written in UCL as defined in the UCL Reference Manual [2.4.3] with restrictions defined in chapter 4.13.4 (description of end item types EGSE–xxx–DERIVED_VALUE).

In contrast HLCL command sequences must be written in the High Level Command Language (HLCL) as defined in the HLCL Reference Manual [2.4.4].

CPL scripts must be written in the Crew Procedure Language (CPL) as defined in the CPL Reference Manual [2.4.6]. The Editor’s CPL Compiler generates CPL scripts in on–board format (also defined in the CPL Reference Manual [2.4.6].

Note that all mentioned languages are very UCL like.

The following sections describe the overall purpose of APs and a description is given how to modify APs. The general editor’s facilities are described in chapter 5.2.5.3. Processing automated procedures can be regarded as a typical example for using the CLS Editor. Items of other end item types are modified in a similar way so that only the differences to developing APs with the CLS Editor are described in further subsections.

The CLS Editor is in fact not only an editor, but an integrated development environment, which also allows for other tasks like compilation, debugging and reporting.

5.2.5.2 Purpose of APs

An automated procedure (AP) constitutes a low level automated control of the subsystem and payload of space system configurations. The objective of an AP is to execute particular subsystem or payload functions. Thus an AP is a computer program or program component, performing a sequence of operations that would otherwise be executed by a human operator. It contains verifiable, reusable processing statements necessary to perform specific operations. Normally APs will remain fixed and will be executed in the frame of action processing. In contingency cases an AP may be modified by the flight crew or ground personnel.

The afore described AP is usually called Flight Automated Procedure (FLAP). The second kind of AP is known as Ground Automated Procedure and is used during the integration and check-out phases of sub-sys-

Copyright per DIN 34

tems in order to allow test executions automation. In this case the command sequences of such an AP have to be regarded as test procedure steps (see section 7.3.1.2 for more information).

APs are written in **User Control Language (UCL).** Although the UCL basic syntax is that of a general–purpose programming language (derived from Modula–2), UCL is a dedicated test and operations language for monitoring and control of spacecraft sub-systems. UCL is a procedural language representing the set of all commands or instructions that can be predefined and stored, beside in APs, also in user libraries (end item type: UCL_USER_LIBRARY) in the Mission Database (MDB).

A third End Item type called UCL_SYSTEM_LIBRARY represents a system library, which provides the basic set of functions and procedures for AP development. These libraries are fixed and shall not be modified by users, but can be used for the purpose of re-compilation.

5.2.5.3 How to Develop APs

The first task to undertake is the creation of an end item of type UCL_AUTOMATED_PROCEDURE or UCL_USER_LIBRARY. This can only be done within CDU versions, which are of one of the domains CGS, EGSE or UCL_LIBRARY as described in Appendix–6.6.4. Thus the procedure for the AP development is as follows:

Creating an Automated Procedure

- **Navigate** to a user tree node as described in section 5.1.2.2.
- *Remember that the CDU version you selected is of domain CGS, EGSE or UCL_LIBRARY.*
- **Select *File*→ *Create Node...*** in the command part of the main window. This opens the ***Create user tree node*** box (see Figure 5–6).
- **Enter** the <*AP–name*> behind the entry ***Name*.**
- **Click** on ***Type...***. This opens the ***Node type list help*** box listing all end item types associated to the CDU domain. For example in Figure 5–8 the domain is CGS, where you can select the type UCL_AUTOMATED_PROCEDURE.
- **Click** on the a.m. end item type and click on the ***Apply*** button in the ***Node type list help*** box. The node type 'virtual' in the ***Create user tree node*** box is now replaced by UCL_AUTOMATED_PROCEDURE.
- **Enter** a Configuration Item number, if applicable. (Configuration Items (CI) are used to identify system components formally to be delivered.
- **Click** on the ***Owner...*** button. This will provide the ***User List*** box, listing all known user of the MDB instance.
- In the ***User List*** box, **click** on the user to be selected as owner followed by a **click** on the ***Apply*** button. Now the user name is displayed behind the ***Owner...*** button in the ***Create user tree node*** box.
- **Enter** some <*description test*> in the field ***Description***.
- **Click** on the ***Apply*** button of the ***Create user tree node*** box. Now the latter box disappears and the AP is listed as part of the Node-List in the I_MDB Navigator window.

The above procedure works the same way for all database items that can be processed with the CLS Editor. Only the operation set for items of other end item type may differ from the operation set for automated procedures. Figure 5–10 shows that, apart from the common operations, the only available operation on an automated procedure is the invocation of the CLS Editor and compiler.

Copyright per DIN 34

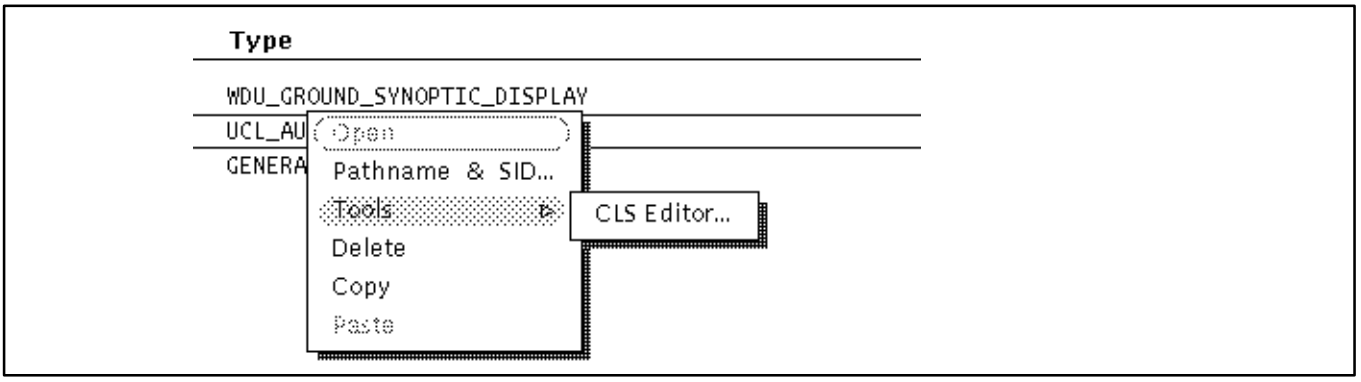


Figure 5–10 : *Operations on End Items of Type Automated Procedure*

The following procedure describes how to access an automated procedure. Database Items of other types are accessed in a similar way. Major differences for them are described below.

Accessing Automated Procedures

- **Start** the ***I_MDB Navigator*** as described in section 5.1.2.1 in procedure ”Starting a Mission Configuration Session”.
- **Navigate** to the automated procedure you want to edit (see 5.1.2.2).
- **Select** the automated procedure and **press** the right mouse button (a menu appears).
- **Select** ***Tools*** → ***CLS Editor*** and **release** the mouse button. After a few seconds the CLS Editor appears on the screen (see Figure 5–11).

An alternative CLS Editor invocation is to double click an CLS processable end item.

The I_MDB tool will display a message that the CLS editor has been started in *batch mode*. This means that you can work with I_MDB and one ore several CLS editors in parallel. I_MDB will **not** go into a busy state and wait until you have finished your work with the CLS editor.

After the above mentioned actions the CLS Editor will come up with its main window. How this looks like is shown in Figure 5–11.

Copyright per DIN 34

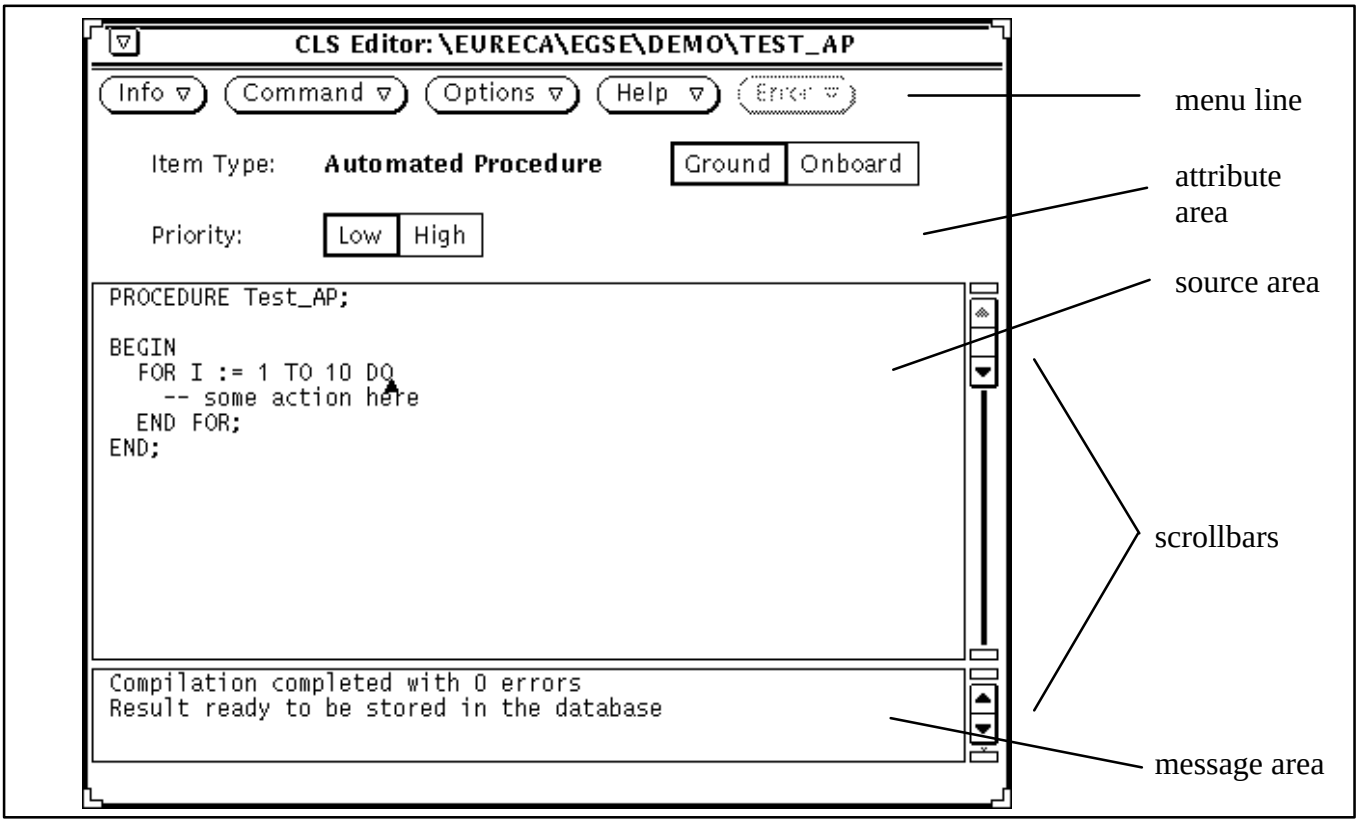


Figure 5–11 : *CLS Editor with UCL Compiler*

Now you can work with the CLS Editor and generate or modify the required AP. The general editing facilities in the source area work very similar to the normal Open Windws ”textedit” editor.

Compiling an Automated Procedure

- **Click** with the **left mouse button** on the ***Command*** button.
- Or **press** the ***Command*** button with the **right mouse button** and select ***Command*** → ***Compile***.

Compilation messages are displayed in the message area.

If there are compilation errors the error button is enabled. Using the error menu you can localize the erroneous source text positions. The text caret is positioned at an error location and the appropriate message is displayed on top of the message area (see Figure 5–12).

Copyright per DIN 34

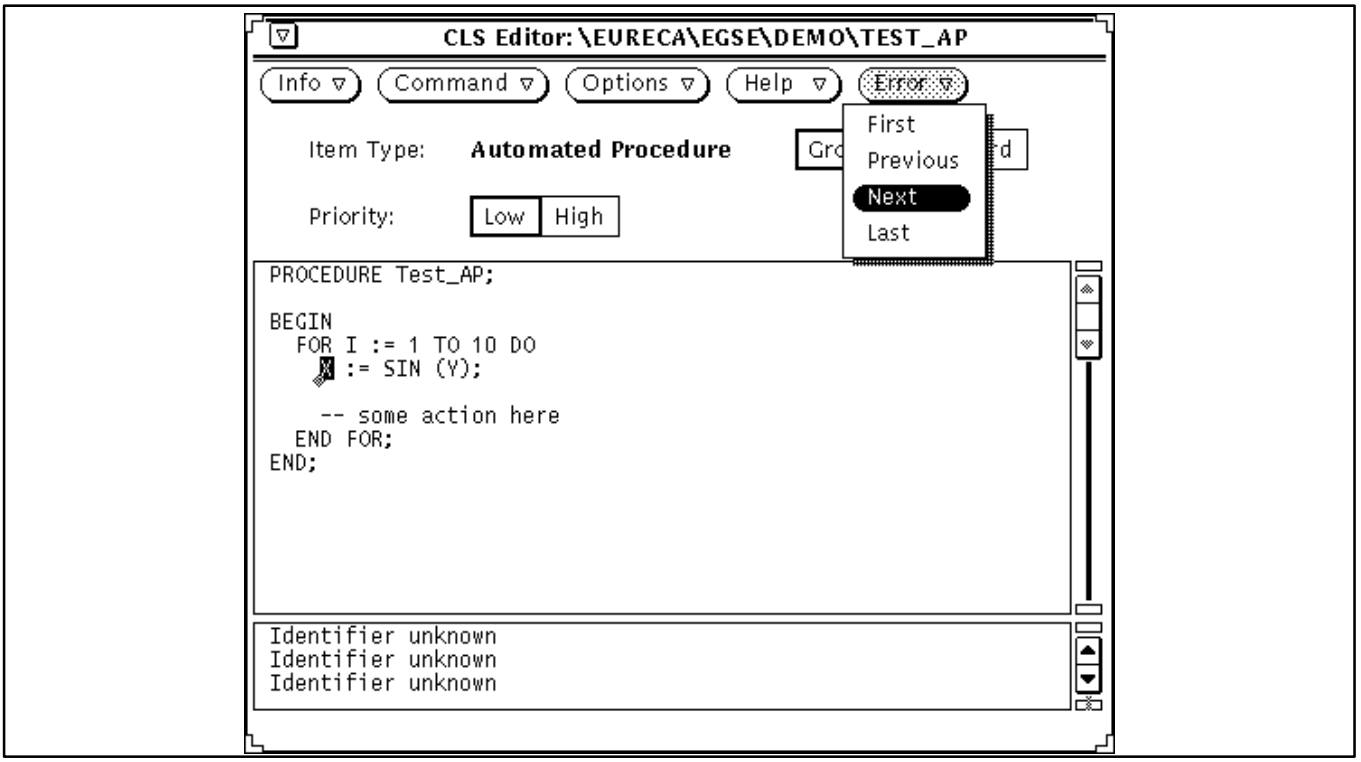


Figure 5–12 : *CLS Editor: Error Menu*

Finding Error Locations

- **Click** with the **left mouse button** on the ***Error*** button to get to the next error location.
- Or **press** the ***Error*** button with the **right mouse button** and select ***Error*** → ***Next***.
- To localize the first, previous or last error **press** the ***Error*** button with the **right mouse button** and select ***Error*** → ***First***, ***Error*** → ***Previous*** *or* ***Error*** → ***Last***.

You can control the compiler's behaviour in some way. For this purpose use the ***Options*** menu. First of all you can generate a compiler listing.

Generation of a Compiler Listing

- **Click** with the **left mouse button** on the ***Options*** button.
- Or **press** the ***Options*** button with the **right mouse button** and select ***Options*** → ***Listing...***

The CLS editor's listing window will come up. It is shown in Figure 5–13.

Copyright per DIN 34

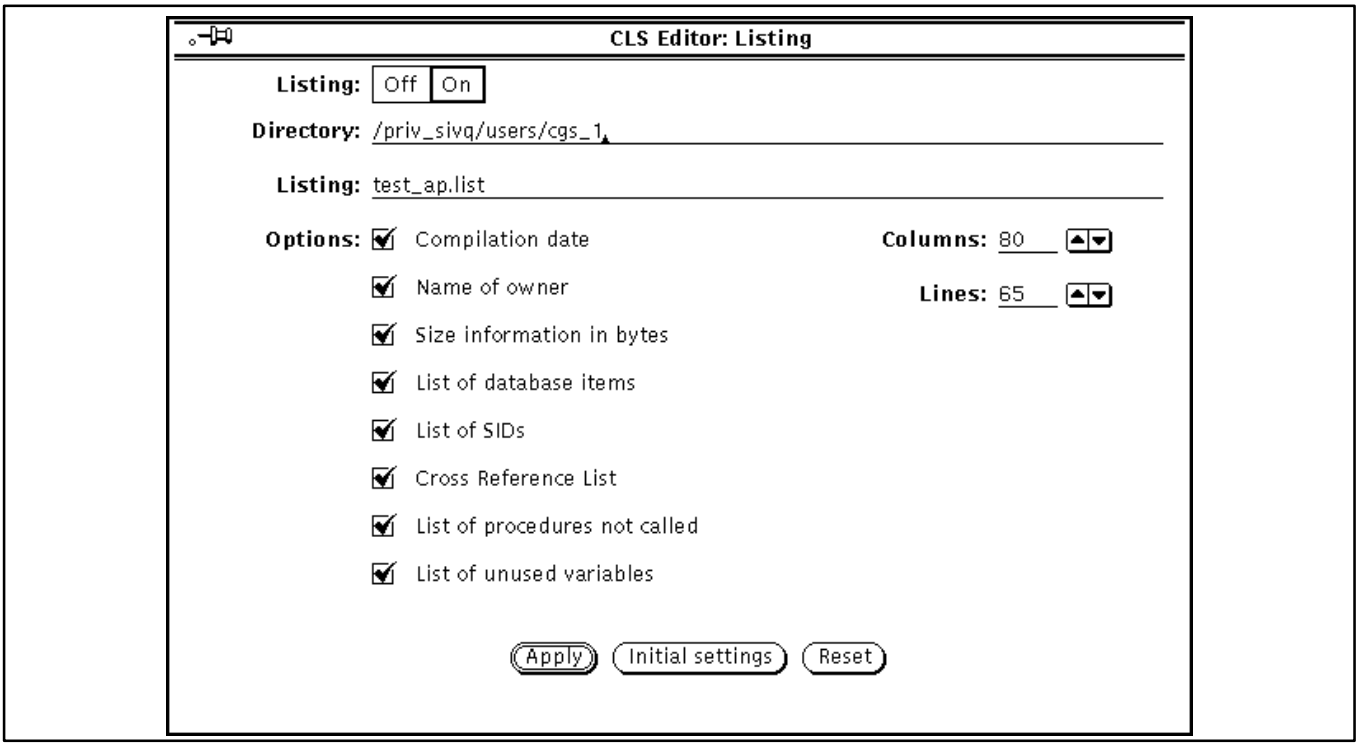


Figure 5–13 : *CLS Editor: Listing Window*

Several listing properties can be set in the listing window. To really get a listing the ***Listing*** choice on top of the listing window must be switched to *on* – if it is not already on – and the changes must be applied with clicking on the ***Apply*** button. A second possibility is to click on the **Initial Settings** button to get the listing default values. The listing window in the figure is one with its initial default values. After clicking on the **Initial Settings** button (which only lets the listing window display the default values) you must click on the ***Apply*** button to make the default values applicable. The third possibility is to click on the ***Reset*** button. This action restores the last "applied" values. Again, click on the ***Apply*** button to make them applicable.

Note that for each (from I_MDB invoked) CLS Editor the listing window's initial values are the default values.

As further option the CLS Editor provides a make mode in which the current compilation unit (the AP) and all compilation units (e.g. UCL user libraries) the AP depends on are compiled automatically if necessary.

Make Mode

- **Press** the ***Options*** button with the **right mouse button** and select ***Options*** → ***Make...***

A simple small window is displayed where you can choose between ***on*** and ***off***. The make mode can be performed on three different display levels.

- **Select** one of Show: ***None, Erroneous,*** *or* ***All***

The default is to display a separate editor window for a compilation unit only in case of compilation errors (Show: ***Erroneous***). It is also possible to compile and store the units in background (Show: ***None***). In this case it is necessary to generate an error list file (see below) to get compilation error messages. The third possibility is to step through all units that have to be compiled (Show: ***All***). For each unit to be compiled a separate editor window comes up and the compilation has to be done manually by selecting ***Command*** → ***Make*** (or

Copyright per DIN 34

clicking with the left mouse button on the Command button). In make mode this entry replaces the ***Command → Compile entry***. Store the item afterwards and quit the window. A new editor window will come up with the next unit or the compilation will continue with the previous unit. During compilation of units other than the starting unit the base editor window is deactivated so that even in this mode a correct compilation order is ensured.

Note that in case of cyclic dependencies between units (e.g. AP A calls B and B calls A) the make process terminates with an appropriate error message.

The CLS Editor can be forced to generate an error list file containing essential information about the compilation process in a short form. The file's contents is the compilation unit name and a list of compilation error messages (in make mode for each compiled unit).

Generation of an Error List File

- **Press** the ***Options*** button with the **right mouse button** and select ***Error List...***

The compiler can be run with or without optimization.

Choosing the Compiler's Optimization Mode

- **Press** the ***Options*** button with the **right mouse button** and select ***Options → Optimize...***

A simple small window is displayed where you can choose between ***on*** and ***off***. The default is *optimization on*.

At any time you can store the automated procedure which you are processing.

Storing an Automated Procedure

- **Press** the ***Command*** button with the **right mouse button** and select ***Command → Store***.

If your compilation was not successful and you try to store the automated procedure, the CLS Editor will ask you for confirmation because in this case only the source can be stored. If there is already compiler generated data in the database from an earlier successfully compiled AP version the store operation will lead to loss of that data because the old generated data is inconsistent with the current AP source code. Figure 5–14 shows how the confirmation request looks like.

If you don't want to store the item – in this case only the source code – just click on the dismiss button of the confirmation window and the item contents in the database will remain unchanged.

On execution of the store operation all item attributes (as given in the CLS editor's item attribute area) are always stored – independent of the item's compilation state. If you want to store the AP and quit the editor afterwards you can do it by simply selecting ***Command → Store & Quit***.

Copyright per DIN 34

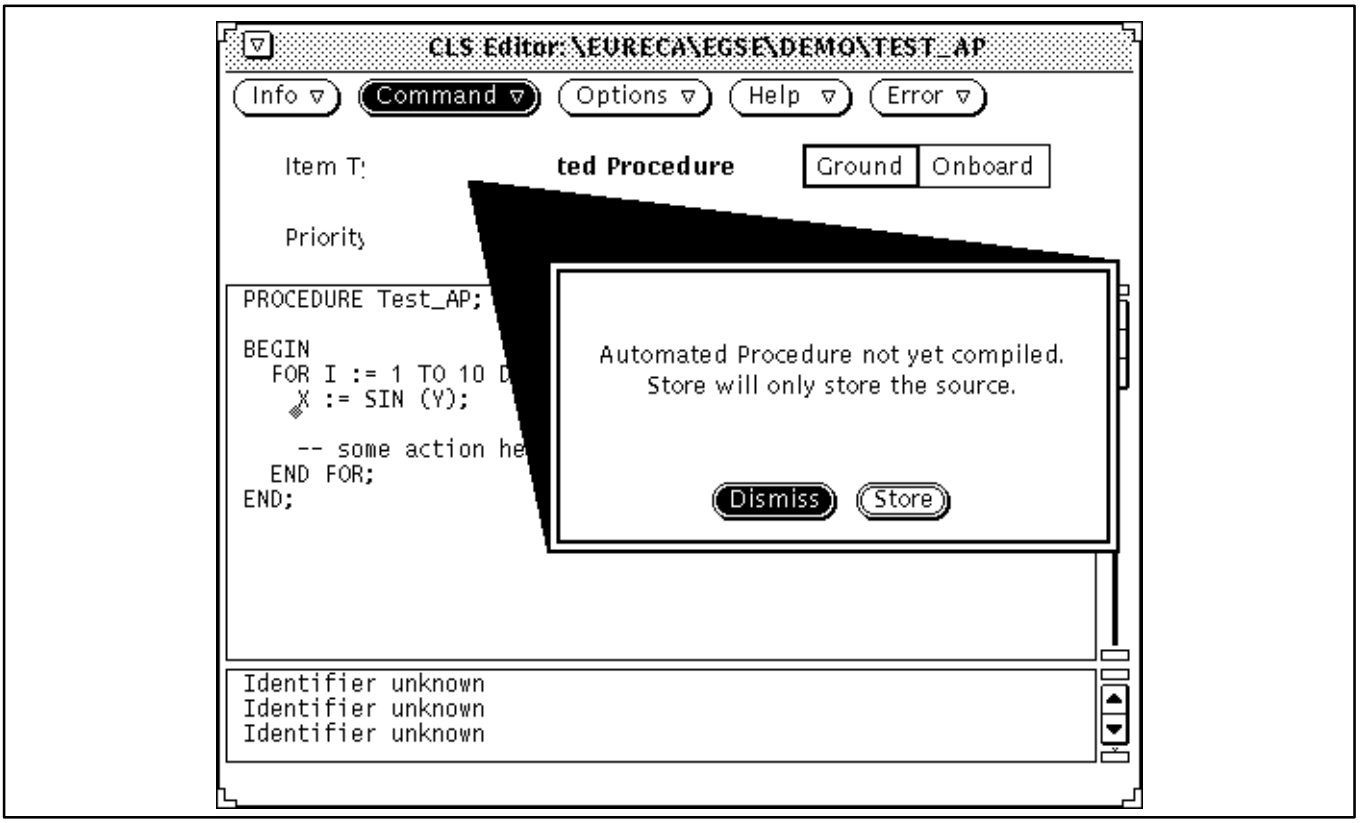


Figure 5–14 : *CLS Editor: Confirm the Store Operation*

While you are working with the CLS editor some history information can be obtained about the user's actions.

Viewing the Action Log

- **Press** the ***Comamnd*** button with the **right mouse button** and select ***Command*** → ***View Log***.

A window comes up which lists the store and compile actions performed since editor startup. This window is especially interesting if a unit has been compiled in make mode without generation of an error list file. A list of the processed units can be obtained from here.

Furthermore while you are processing an automated procedure you can obtain some information about it.

Getting Information about an AP

- **Press** the ***Info*** button with the **right mouse button** and select ***Info*** → ***Item***.

An item information window comes up and displays some of the item's attributes, see Figure 5–16.

Copyright per DIN 34

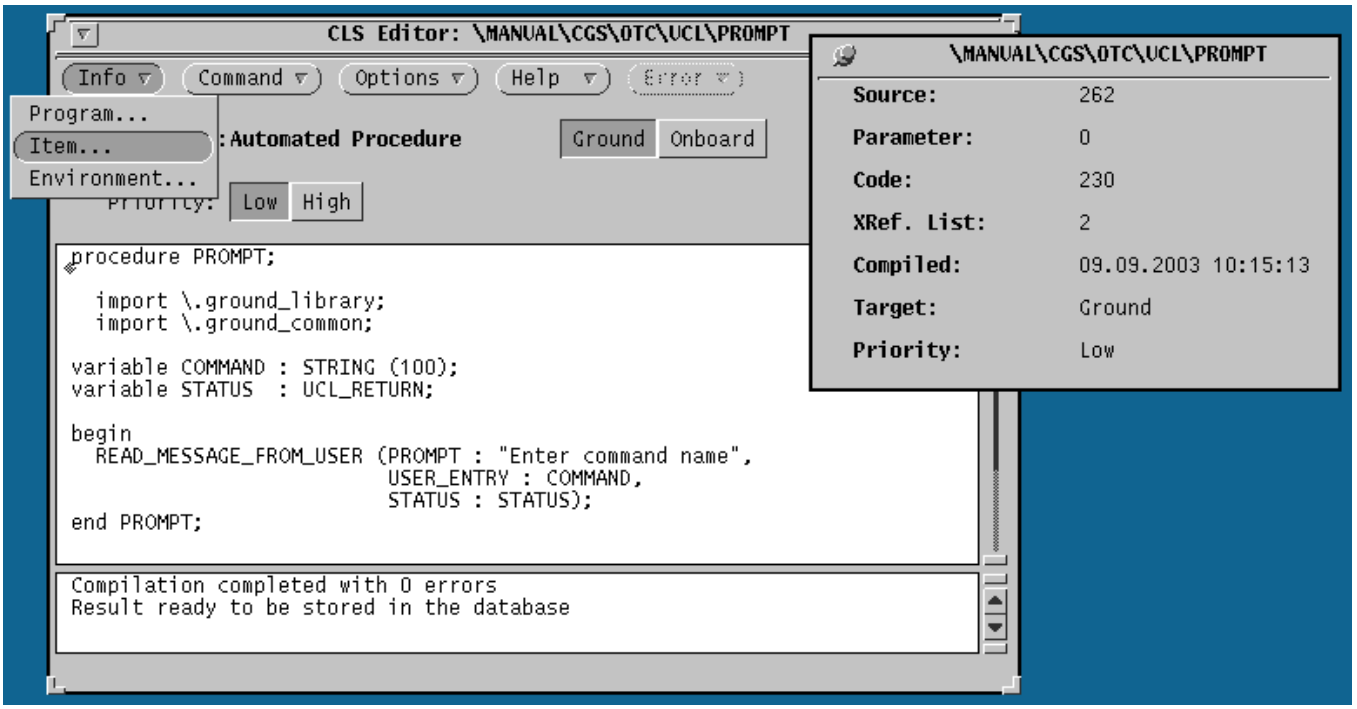


Figure 5–15 : *CLS Editor: The Item Information Window*

The item information window **Info–>Item** for an automated procedure comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of compiler generated code in bytes
- number of entries in the cross reference list
- compilation date
- size of the compiler generated debug table in bytes
- deleted
- priority (also shown in the CLS Editor's main window)

If the item information window is on the screen while you modify the automated procedure the changes are recognized and the information window will be updated within a short period of time (normally one second). You can simply observe this by typing a character in the source area. The character count in the information window will be incremented.

For an initially loaded AP (i.e. an unmodified AP) the item information window shows attribute values reflecting the item contents in the database.

The ***Info*** ***Program*** button can be used to obtain some global information about the CLS editor such as an enumeration of the different compilers it comprises.

The ***Info*** **Environment** button can be used to obtain global database information such as CCU or CDU info.

Copyright per DIN 34

Help on names of end items during work with the CLS Editor can be obtained in the following manner.

Getting Pathname Help

- **Press** the ***Help*** button with the **right mouse button** and select ***Help*** → ***Pathname Support***...

Remark: Pathname Help is not supported in CGS version 6.1

The pathname support window comes up and allows you to look for existing end items. The selected name will be copied into the editor's source text window at the text caret position when clicking on the ***Apply*** button. For *each* pathname to be inserted in the source the action sequence "selecting ***Help*** → ***Pathname Support***, clicking on the ***Apply*** button" must be performed. For a detailed descripton of the pathname support window please refer to the corresponding detailed user manual [2.2.1.3].

When you finished your work want to quit the CLS Editor it checks if you modified the automated procedure but didn't store the changes yet in the database. In this case you will be asked to execute a store operation or to intentionally discard the changes as shown in Figure 5–16. You also have the possibility to cancel the quit operation.

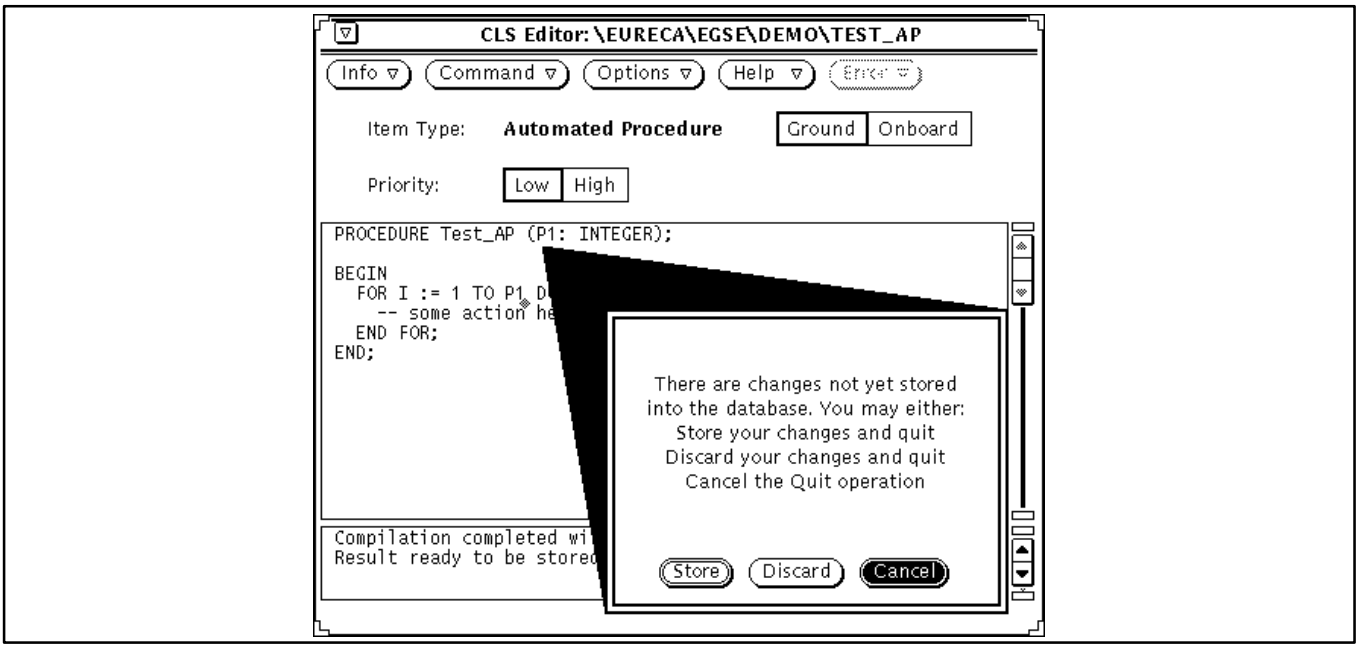


Figure 5–16 : *CLS Editor: Quit — Confirmation Request*

5.2.5.4 Developing a User Library

For processing end items of type User Library as for APs a domain of type CGS, EGSE or UCL_LIBRARY must be chosen. The major difference with respect to the usage of the CLS Editor is that in the source area as one choses the user library specification or the user library body is displayed. You can toggle between them by clicking on the spec/body choice of the CLS Editor's main window, see Figure 5–18. On the left side the editor's window is shown with the specification loaded and on the right side the editor has loaded the user library body.

Copyright per DIN 34

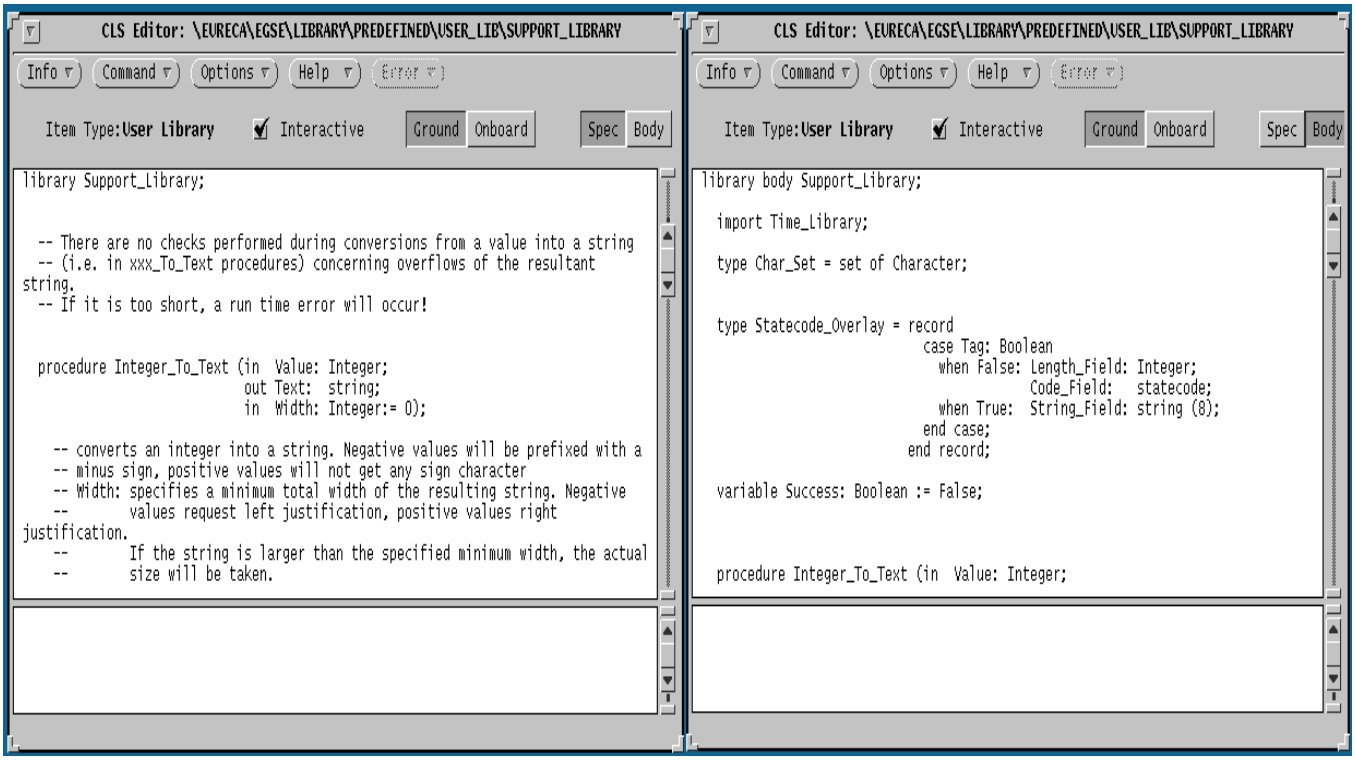


Figure 5–18 : *CLS Editor: Processing a User Library*

The specification and the body source are processed in a similar way to automated procedures. But when you try to compile the user library source the editor will force you to consider the correct compilation order. If you try to compile a user library body and the specification is not already compiled you will get a hint to please compile the specification first.

The item information window for a user library comprises the following attributes:

- size of the specification and body source code in characters
- size of compiler generated code in bytes
- size of compiler generated symbol table in bytes, generated during specification compilation
- size of the compiler generated debug table in bytes, generated during body compilation
- number of entries in the specification and body cross reference list
- compilation date for screen and body
- target (ground or space)
- interactive flag (also shown in the CLS editor's main window)

5.2.5.5 Developing a System Library

The CLS editor's usage for system libraries is nearly the same as for automated procedures.

Here again only *one* source is loaded into the source area, i.e. the specification of the system library.

Copyright per DIN 34

The item information window for a system library comprises the following attributes:

- size of the source code in characters
- size of compiler generated symbol table in bytes
- number of entries in the cross reference list
- compilation date
- target (ground or space)
- interactive flag (also shown in the CLS Editor's main window)
- system library ID (also shown in the CLS Editor's main window)

The system library's ID depends on the implementation of the system library body. It must be set correctly to calls to system library procedures work correctly. The default value 0 is an invalid ID and if you try to store the system library specification with this ID you will be asked for confirmation.

5.2.5.6 Developing an HLCL Sequence

An HLCL Command Sequence may comprise simple HLCL commands and HLCL compound statements such as if-then-else or loops. A detailed description is given in the HLCL reference manual in document 2.4.4. HLCL is syntactically nearly the same as UCL. For this reason the HLCL reference manual is an extension to the UCL reference manual.

HLCL sequences can be stored in files (see also *Invocation Interface*, chapter 5.2.5.10 below) or in the database. The latter ones are processed with the CLS editor in a very similar way to automated procedures, but for HLCL sequences only a domain of type CGS or EGSE can be chosen. The end item type name is HLCL_COMMAND_SEQUENCE.

Two types of HLCL sequences can be distinguished: open sequences and procedure sequences.

For open HLCL sequences the operation *compile* has a different meaning. HLCL sequences are executed with the HLCL interpreter which accesses the source code. So no compiler generated code is necessary. But nevertheless you may *compile* an HLCL sequence in the sense that the sequence is **checked** for syntactical errors.

An HLCL check is significantly fewer than a UCL compilation. E.g. no checks are made (and cannot be made) if used pathnames are really names of existing end items. Also no checks can be made if used variables are declared. Variables are rather variables of an interactive HLCL session than local objects of a sequence. So it is sufficient to declare them immediately in an HLCL session before executing the sequence.
If objects used in an HLCL sequence are not defined when you *execute* the sequence you will receive error messages from the HLCL interpreter.

The item information window for an HLCL sequence comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of "compiler generated code" in bytes, i.e. information whether the sequence has been checked or not. (0 for not checked, 1 for checked)
- check date

Creation of an HLCL Sequence in the Unix file system:

To create or edit an HLCL Command Sequence in the Unix file system, follow these steps:

Copyright per DIN 34

Create HLCL Command Sequence in Unix file system

- Call the CLS Editor from a Unix shell:
  $CLS_HOME/bin/sun5/cls_editor <*filename*>.hlcl <*options*> (SUN only)
  $CLS_HOME/bin/linuxi/cls_editor <*filename*>.hlcl <*options*> (LINUX only)
- If the sequence file does not exist the CLS Editor asks for a creation confirmation, see Figure 5–19.
- Answer OK if you want to create the file or Cancel, if you want to quit the CLS Editor immediately.

See also the CLS Editor invocation interface in 5.2.5.10!

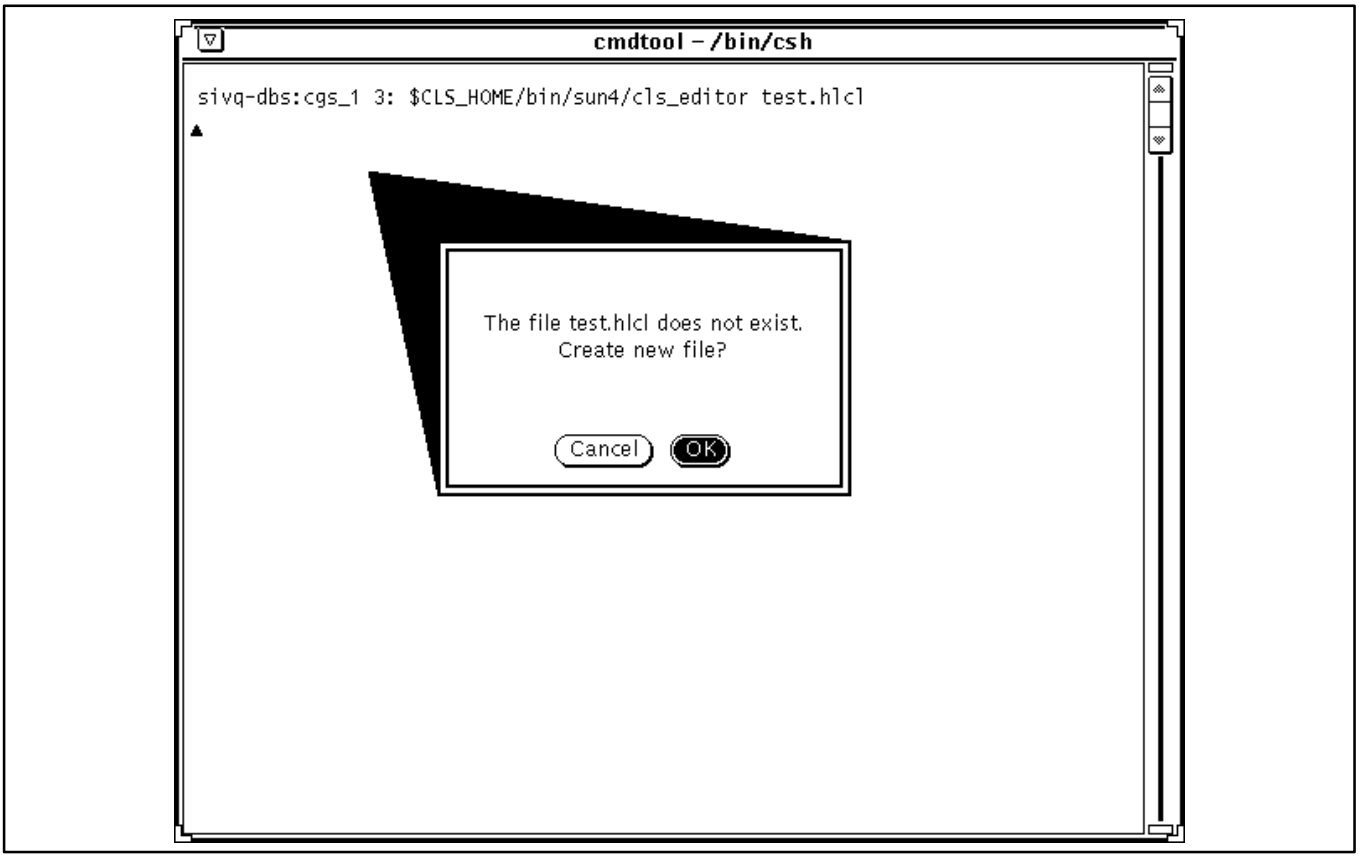


Figure 5–19 : *CLS Editor Asking for Confirmation of File Creation*

Edit HLCL Command Sequence in Unix File System

- After confirmation or if the sequence already exists, the CLS Editor window pops up.
- Now you may enter or edit the HLCL Command Sequence the same way as any editing is performed in the CLS Editor.
- If you want to store your changes in the file, leave the CLS Editor via the **Command–>Store & Quit** menu option (see Figure 5–20).
- Otherwise, use the **Quit** option of the window menu.

Copyright per DIN 34

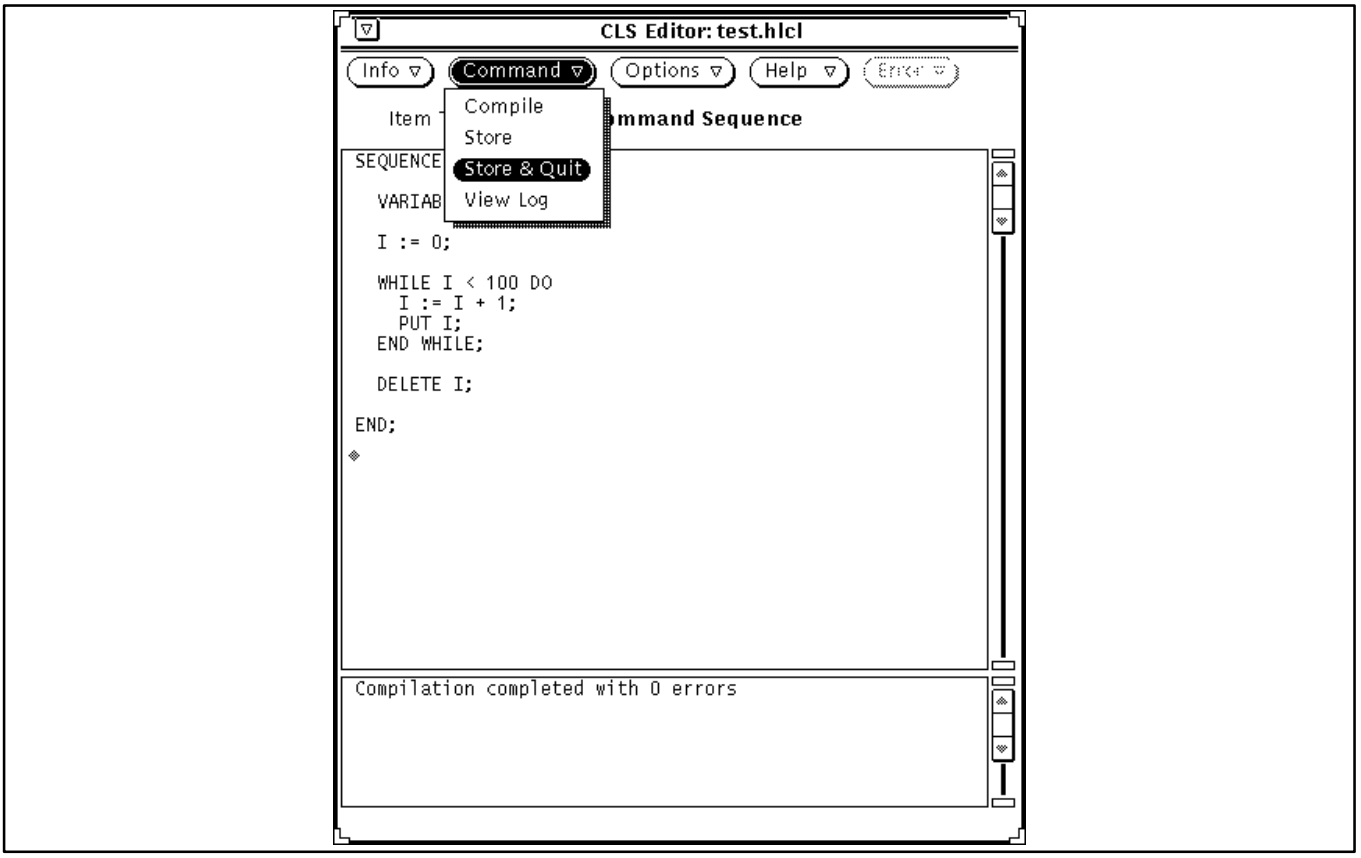


Figure 5–20 : *HLCL Command Sequences in the Unix File System*

5.2.5.7 Developing a CPL Script

The APM onboard LAPTOP provides the capability to execute crew procedures. These procedures can be monitored and controlled by the crew member during execution. They are presented in a readable way so that any step of the procedure is understood by the crew member.

The language which constitutes the presentation of crew procedures to the crew as well as the language of the source code to be developed is the CPL (Crew Procedure Language). The major means how a CPL script influences the behaviour of the APM is the commanding of the software which controls the system.

These commands to the onboard software are called software commands. They are defined in the Mission Data Base and can be used in CPL scripts and other sources like Onboard Synoptic Displays.

The basic entity is the script. It contains elements as defined by the script language (for more information about CPL refer to 2.4.6).

A CGS supported on board script language has four different representations:

- the ”source code” representation is the ASCII text as entered by the developer using the Crew Procedure Language,
- the ”processing code” representation is the result of the compiling process. This is the code given to the onboard interpreter for processing,
- the ”presentation code” is a subset of the script. This is what the crew will see during crew procedure operations,

Copyright per DIN 34

- the logic flow diagram (LFD) shows the script as a flow diagram where specific language constructs are replaced by graphical items. Remaining text is displayed as ”presentation” code.

CPL scripts are processed with the CLS Editor in a very similar way to automated procedures. For processing end items of type CPL script a domain of type CGS or SDDF must be chosen.

The item information window for a CPL script comprises the following attributes:

- size of the source code in characters
- size of compiler generated code in bytes
- number of entries in the cross reference list
- compilation date

5.2.5.8 Developing an Item with Parameter List

The CLS Editor’s usage for items which have parameter lists is nearly the same as for automated procedures.

The item information window for these items comprises the following attributes:

- size of the source code in characters
- number of parameters
- size of compiler generated symbol table in bytes
- number of entries in the cross reference list
- compilation date
- target environment (ground, on board)

The parameter lists of some items must obey some constraints. Please refer to Chapter 6.6.4 and document [2.4.2] for a description of these constraints.

If the CLS Editor is invoked for a newly created item it will generate source for an initial parameter list and display it in the source window. In general this created source represents an empty parameter list, i.e. just a pair of parenthesis. But if the parameter list constraints are so rigid that they lead to a fixed parameter list form, source for a non empty parameter list obeying the given constraints is generated. E.g. the latter case applies if the parameter list constraints only allow one parameter, which must be of mode in, of type real and without default value.

5.2.5.9 Developing an Expression for Derived Values

The CLS Editor’s usage for derived value items is nearly the same as for items with parameter lists.

Expressions are used to describe the calculation of values from other values (end items of type EGSE_XXX_MEASURMENT, EGSE_xxx_SW_VARIABLE or EGSE_xxx_DERIVED_VALUE).

The expressions must follow UCL syntax and obey some constraints. Please refer to Chapter 6.6.4 for a description of these constraints.

The item information window for these items comprises the following attributes:

- size of the source code in characters
- size of compiler generated code in bytes
- number of entries in the cross reference list (i.e. the number of enditems used in the expression)
- number of entries in the SID list (i.e. number of SIDs used in the generated code)
- compilation date

Copyright per DIN 34

5.2.5.10 CLS Editor Invocation Interface

The CLS Editor is a UNIX process named `cls_editor`, located in directory `$CLS_HOME/bin/<architecture>`. It is to be started with command line parameters and options. The process returns a completion code. In interactive mode the completion code is always 0. In batch mode it indicates compilation success/failure:

0 = no compilation errors,
1 = compilation errors.

Depending on its type, a compilation unit may be contained in the database or in a file (the file may be a named pipe). When contained in the database, the attributes of the database item determine the type of compilation unit. When contained in a file, the type of compilation unit is, by default, determined by the file name suffix:

`.hlcl`	HLCL command sequence
`.fwdu`	FWDU command table

If the file name suffix does not uniquely define the type, it must be explicitly given with the `–unit` option.

Option names and keywords are case–insensitive. They may be uniquely abbreviated. Compound names may also be abbreviated by abbreviating the single components (e.g. `–error_list` may be abbreviated as `–err_li` or `–err`).

Call:

```
cls_editor <pathname> <options>     (call for database item)
cls_editor <filename> <options>     (call for file)
```

Options:

`–help [...]`

displays a short help information of parameters and options, adding the value `"..."` shows long format:

```
cls_editor –help        short help format
cls_editor –help ...    long help format
```

`–environment <environment>`

defines the database user environment. If omitted, the environment will be obtained from the environment variable `MDA_ENVIRONMENT` in the same syntactic form:

```
<environment> ::=
  CCU <element_config> <mission> <system_tree_version> <ccu> |
  CDU <element_config> <mission> <system_tree_version> <cdu>

<ccu> ::=
  <pathname> <ccu_name> <version>.<issue>.<revision>

<cdu> ::=
  <pathname> <version>.<issue>.<revision> <test_version> <instance>
```

Example:

```
–env "CCU APM DUMMY_MISSION 1 \APM\TEST\GROUND\CLS CLS_TEST 1.0.0"
```

Copyright per DIN 34

-unit <unit>

defines the type of compilation unit contained in a file. If omitted, the type will be determined by the file name suffix (see above).

```
<unit> ::= HLCL_SEQUENCE | FWDU_COMMAND_TABLE
```

-file
-database

For the very rare case that the name given as a parameter cannot be uniquely identified as a file name or database pathname (e.g. file name starting with '\'), one of these options can be used to resolve the ambiguity.

-make

requires that the specified compilation unit as well as all units referenced by this unit directly or indirectly are to be updated. Items will be compiled, if necessary.

-forced

requires the item and all referenced items to be recompiled, regardless of their consistency status.

-fast

requires that only the specified compilation unit is to be updated. Referenced items are checked for up–to–date status, but not recompiled.

-batch

indicates that the CLS Editor is to be called in batch mode, no window will appear

-error_list <filename>

requires an error list to be generated in a file. For each compiled unit it contains a section of the form

```
<file_or_path_name>
<line> <column> <error_message>
...
<line> <column> <error_message>
```

The first line contains the path name or file name, resp., of the compiled unit. It starts in the first column of the the line. Each of the following lines contains a single message prefixed with the line and column number as a reference to a position in the source text. These lines are indented with two spaces. The lines are sorted by line and column numbers. If line and column are both 0, the message does not refer to the source code but reports some general event, e. g. an internal compiler problem.

For user libraries separate sections are generated for specification and body, the text "spec" or "body" will be appended to the corresponding path name header. For a compilation with the `-make` option one section per compiled unit will be generated. Sections are separated by an empty line.

If the file name is omitted, it will be derived from the input file name or the database item node name, resp.:

Copyright per DIN 34

`-listing [ <filename> [ <filename> ] ]`

requires a listing to be generated in a file. For user libraries separate listings are generated for specification and body. If file names are omitted, they will be derived from the input file name or the database item node name, resp.:

`<name>_.list` for the specification
`<name>.list` for the body (and all non–library units)

Parameter/Option Combinations:

	parameter	–environment	–make –fast	–unit	–file	–database	–batch	number of files –error_list	number of files –listing
AP	<path>	x	x			(x)	x	0 – 1	0 – 1
User Library	<path>	x	x			(x)	x	0 – 1	0 – 2
System Library	<path>	x	x			(x)	x	0 – 1	0 – 1
HLCL Sequence	<path> <file>	x x		x x	 x	x	x x	0 – 1 0 – 1	0 – 1 0 – 1
CPL Script	<path>	x	x			(x)	x	0 – 1	0 – 1
FWDU Cmd. Table	<file>	x		x	(x)		x	0 – 1	0 – 1
Parameter List	<path>	x	x			(x)	x	0 – 1	0 – 1
Derived Value	<path>	x	x			(x)	x	0 – 1	0 – 1

x = option allowed
(x) = option allowed but meaningless (redundant)

Figure 5–21 : *Parameter Option Combinations for Various Compilation Units*

Copyright per DIN 34

6 TEST AND CONTROL PREPARATION

Before beginning with test preparation you should first familiarize yourself with chapters 6.1 and 6.2 regarding the concept and use of the MDB tool.

For a more detailed information, one should also read the mission database documentation!

6.1 Creating a New MDB Configuration

This chapter explains how to create a new MDB configuration before running a test. A description is provided of all data to put in the MDB which are mandatory to make a configuration running as a test.

Note that certain information is most likely already in the database and just needs to be referenced in the CCU. This applies specially for HK_VALUES, Test configuration, User definitions, Workstations and EGSE node definitions.

6.1.1 Organising the Data

The structuring of data – specially the partitioning in CDUs – is used to minimize the redundant creation of data and to maximize reusage of data by referencing.

The following end items should be separated out, i.e. for each category an individual CDU should be created.

* Test execution node, database node, simulation node and participating workstation node end items
* Software Variables
* Test configuration end items
* UCL System Libraries
* Ground symbol library in case Synoptic Displays are using Symbols
* deleted

The organisation of the user data should be done considering

- meaningful structuring of the tree, allowing to handle groups of data as a virtual subtree. This is especially applicable for measurements/sw_variables/derived values as well as ADUs and GDUs.
- management aspects with respect to version control and import/export handling (don’t mix data from different development groups or life cycles in one CDU)
- avoidance of spreads over different CDUs where possible (group data referencing eachother together). This makes it easier to get to consistent data subsets.
- separate test data from operational data

To generate a certain test configuration a CCU has to be created which references all CDUs that are necessary to define the test configuration.

Copyright per DIN 34

6.1.2 System Prerequisites

Before you start with test preparations take a short look at the following sections. The files SYSTEM_TOPOLOGY_TABLE and the CGS athorisation concept contain certain restrictions. The HW and the processes running on the HW are defined here and the user rights are given.

6.1.2.1 The SYSTEM_TOPOLOGY_TABLE

The SYSTEM_TOPOLOGY_TABLE is maintained by the **CGS administrator** only.
CGS provides the script `change_system_topology` which allow several tasks to modify the system topology table.

Following features can be changed in the system topology table:

- `Add new workstation`
- `Add new test node`
- `Add processes to a known host`
- `Remove some processes for a host`
- `Remove a host completely`
- `Remove a process completely`

The changed SYSTEM_TOPOLOGY_TABLE is then created automatically.

```
#
#-------------------------------------------------------------
# An example SYSTEM_TOPOLOGY_TABLE is given below:
# In general, this example will not be accepted on your
# infrastructure, because all hostnames must be defined
# in your /etc/hosts file.
#-------------------------------------------------------------

SITE:DASA-RI

cgs_1: HCI_01
cgs_1: TSCV_01
cgs_1: TEV_01

vicos_1: HCI_12

svf_5: HCI_09
svf_5: TEV_02

aiv_s: DBS_01
aiv_s: DBS_02
aiv_s: DBS_03

hp847: TES_01
css3s: CSS_01

eureca-1: TES_02
```

Figure 6–1 : *The SYSTEM_TOPOLOGY_TABLE*

Names used in the end items are the CGS internal application names not the UNIX name of the machine. To find the internal names of a workstation, a DB server or a test execution node you have to look into the SYSTEM_TOPOLOGY_TABLE file located in the UNIX directory $CGS_HOME/local/config or use the

Copyright per DIN 34

TSCV Program.
The names on the left side of the colon are the UNIX host machine names (as laid down in the /etc/hosts file after the internet address), the names on the right side of the colon are the internal application (logical) names.

- *To define the end items of type EGSE_NODE correctly the user has to know the logical name of the machines used in the test.*

6.1.2.2 Granting privileges to CGS user

User authorization is required for all CGS user. User authorization is defined by user and system privileges.

- *The CGS user authorizaton concept and tools is decribed in chapter 10.*

Copyright per DIN 34

6.1.3 Creating a New MDB Configuration – Defining DB End Items

There are a number of database enditems which describe the test facility and the configuration of the facility for a given test. This includes definitions of the individual HW nodes of the test equipment, the individual SW entities and the overall configuration.

6.1.3.1 Defining the Node List

Each test configuration has to contain at least three standard node types: a Workstation node, a DB server node and a Test node. These nodes reside on – at least – two computers: one executing the Test node and the other executing the Workstation node and the DB Server node.
As soon as the test configuration is expanded e.g. a Simulation node running CSS models, it is necessary to expand the configuration with more computers.

The node list itself serves as an input list for the definition of test configurations.

The node list consists of end items of type EGSE_NODE.

These end items describe a specific part of the HW of the facility. The main purpose is to provide access to parts of the facility via pathnames from UCL/HLCL level, but the end item also defines the 'role' of the node, e.g. in the EGSE environment it determines whether the node is a workstation, a test node, a database server...

To define an end item of type EGSE_NODE three inputs can be made:

- Node Type – this determines wether the node is a is a workstation, a test node, a database server, a simulation node, a front–end equipment or a UUT.
- Logical Name – this connects the desired HW (see section 6.1.2.1)
- CGS Internal Name – this is an internal marker for the checkout and test system

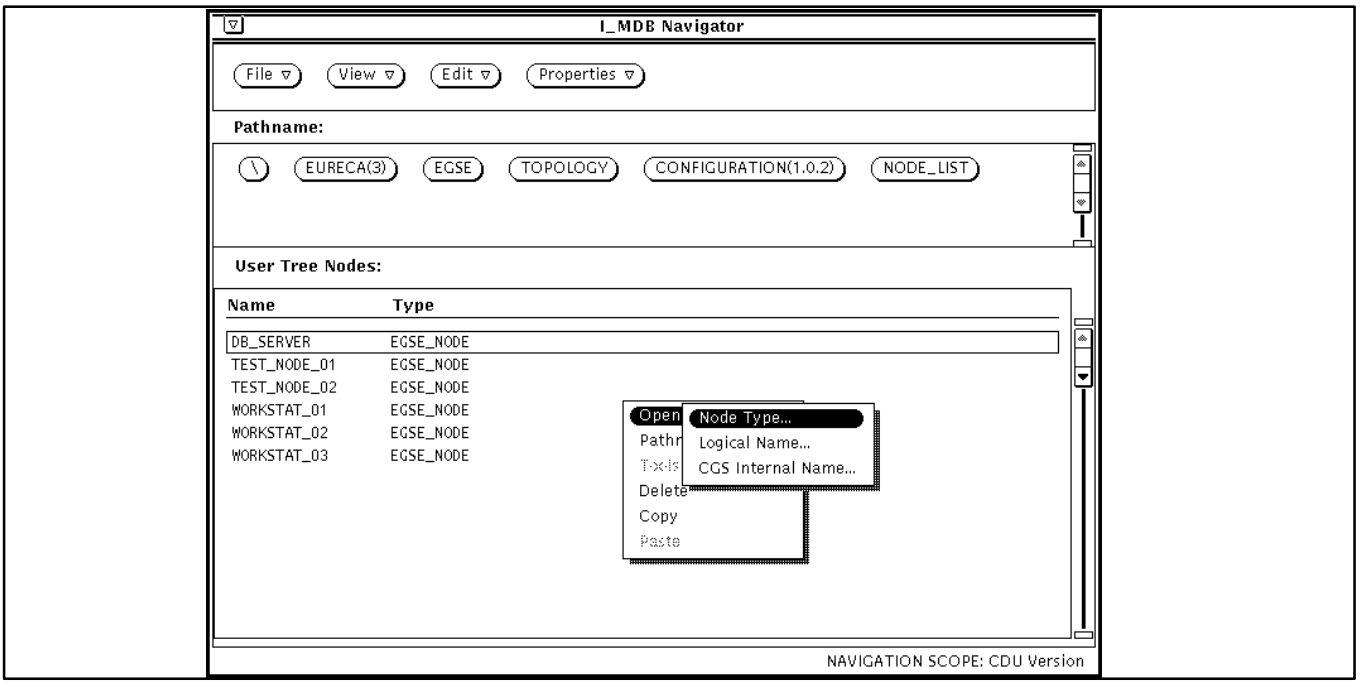


Figure 6–2 : *Each Entry in the Node List can be Defined by Three Attributes – Node Type, Logical Name and CGS Internal Name.*

Copyright per DIN 34

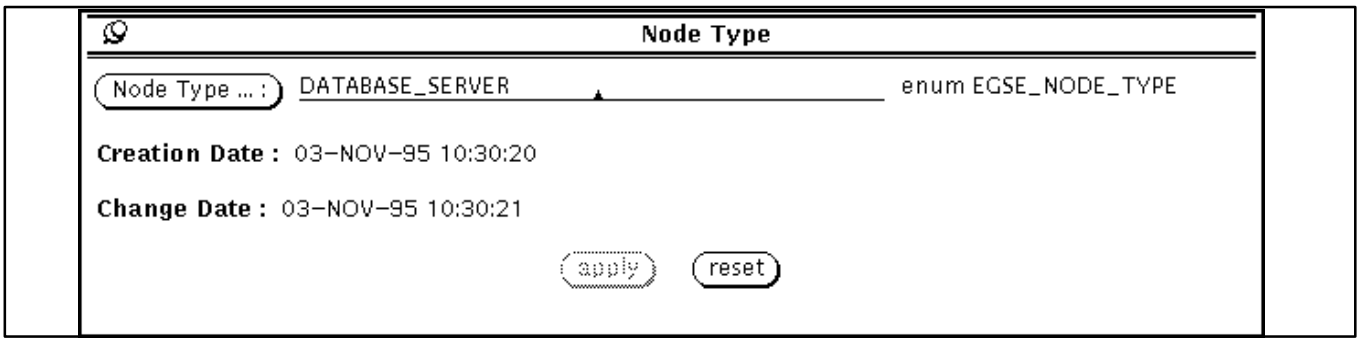


Figure 6–3 : *The Node Type Defines the Function of the Node in the Test Configuration*

For the node types WORKSTATION, SIMULATOR, DATABASE_SERVER and TEST_NODE following rules are valid:

- all three inputs are mandatory
- the logical name is the machine name which can be found in the SYSTEM_TOPOLOGY_TABLE
- a workstation node has the internal prefix HCI
  a simulator node has the internal prefix CSS
  a test node has the internal prefix TES
  a database server has the internal prefix DBS

Note that you can't choose any node type you like.
The assignment (which task may be performed on which computer) is already made in the SYSTEM_TOPOLOGY_TABLE.
Example: Take a look at Figure 6–1. The computer svf_5 *may be used as workstation, the machine* hp847 *as a test node, the* aiv_s *as database server. Note that in the last case it is possible to define three different DB nodes.*

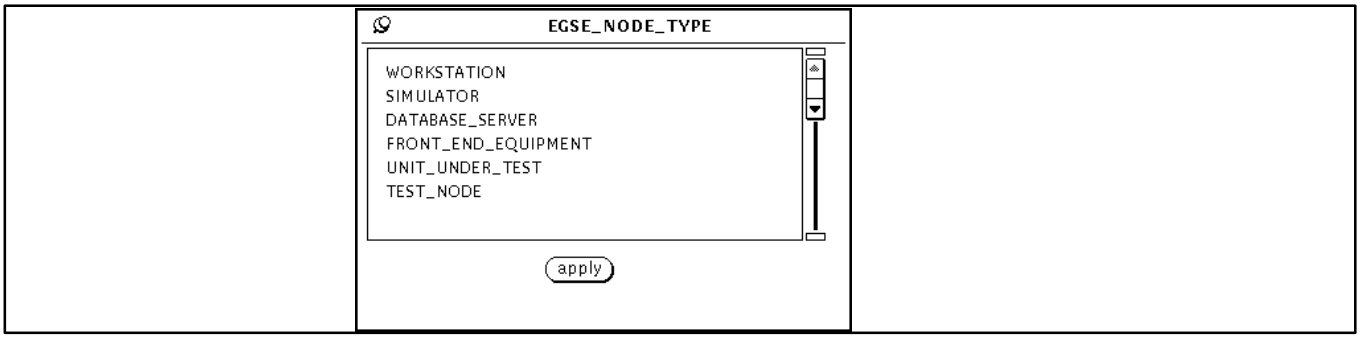


Figure 6–4 : *The Node Type Describes the Function of the Node in the Test Configuration*

For the node types UNIT_UNDER_TEST and FRONT_END_EQUIPMENT the prefix input should be omitted. The logical name depends on the type of HW, i.e. the logical name could be an internet address, a bus address etc.

The following procedure is valid for the four common node types used in a test configuration.

Copyright per DIN 34

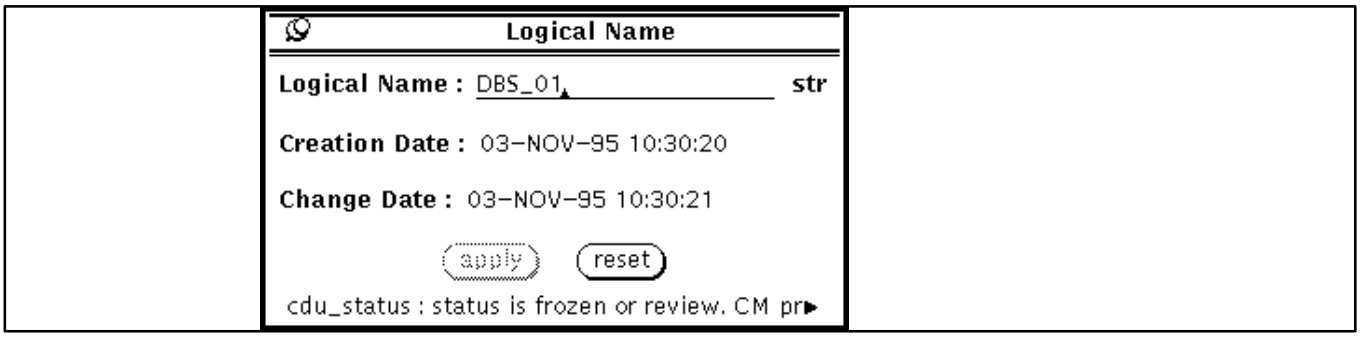


Figure 6–5 : *The Computer with the Logical Name DBS_01 is Linked to the Node Definition*

Defining a DB Server Node

- If not already done, create a CDU (domain: EGSE) for the node definitions.
- *Note that a node is not necessarily a separate computer system !*
- For each node repeat the following steps.
- Create an end item of type EGSE_NODE.
- Select the end item.
- Press the right mouse button and select **Open –> Node Type...**
- The Node Type window appears (see Figure 6–3) Press the **Node Type...** button.
- A selection list window pops up. Select the node type DATABASE_SERVER.
- **Apply** your input.
- Press the right mouse button and select **Open –> Logical Name...**
- Type the logical name of the desired computer in the **Logical Name:** field (see Figure 6–5).
- Press the **apply** button.
- Press the right mouse button and select **Open –> CGS Internal Name...**
- The CGS Internal Name window appears. Press the **CGS Prefix...** button.
- A selection list window pops up. Select the CGS_Node_Prefix type DBS.
- **Apply** your input.

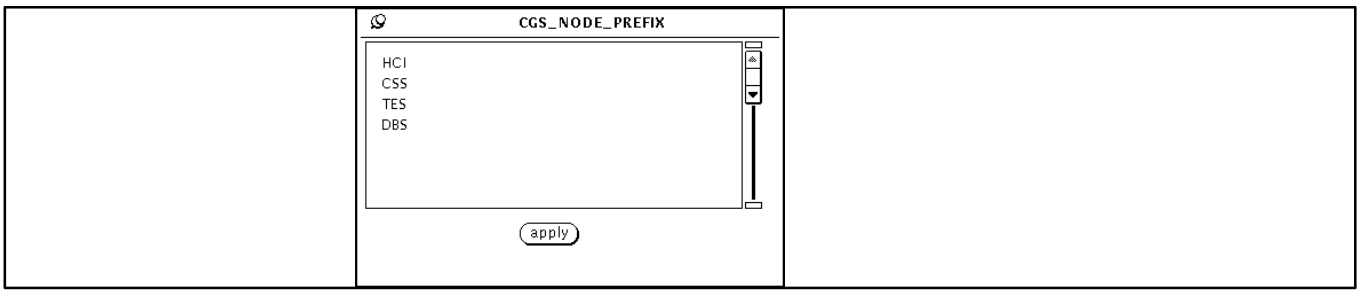


Figure 6–6 : *The Prefix is an Internal Marker*

The node list is later used as a selection list for the definition of a test configuration.

Copyright per DIN 34

6.1.3.2 Defining the Test Configuration

For each test configuration an end item of type EGSE_TEST_CONFIGURATION has to be created.

This end item describes the actual configuration of the facility for a given application. For the checkout and test system, this is the actual configuration of the test equipment to be set–up for a given test.

The test configuration is described in terms of references to other MDB items of type EGSE_NODE and EGSE_SW plus the definition of MDB contents to be 'downloaded' to test nodes via SCOE files.

Interactive type test nodes and service test nodes can be defined such as HCI or CIS test nodes.

End items of type test configuration are later used in the CGS product TSCV to actually do set-up.

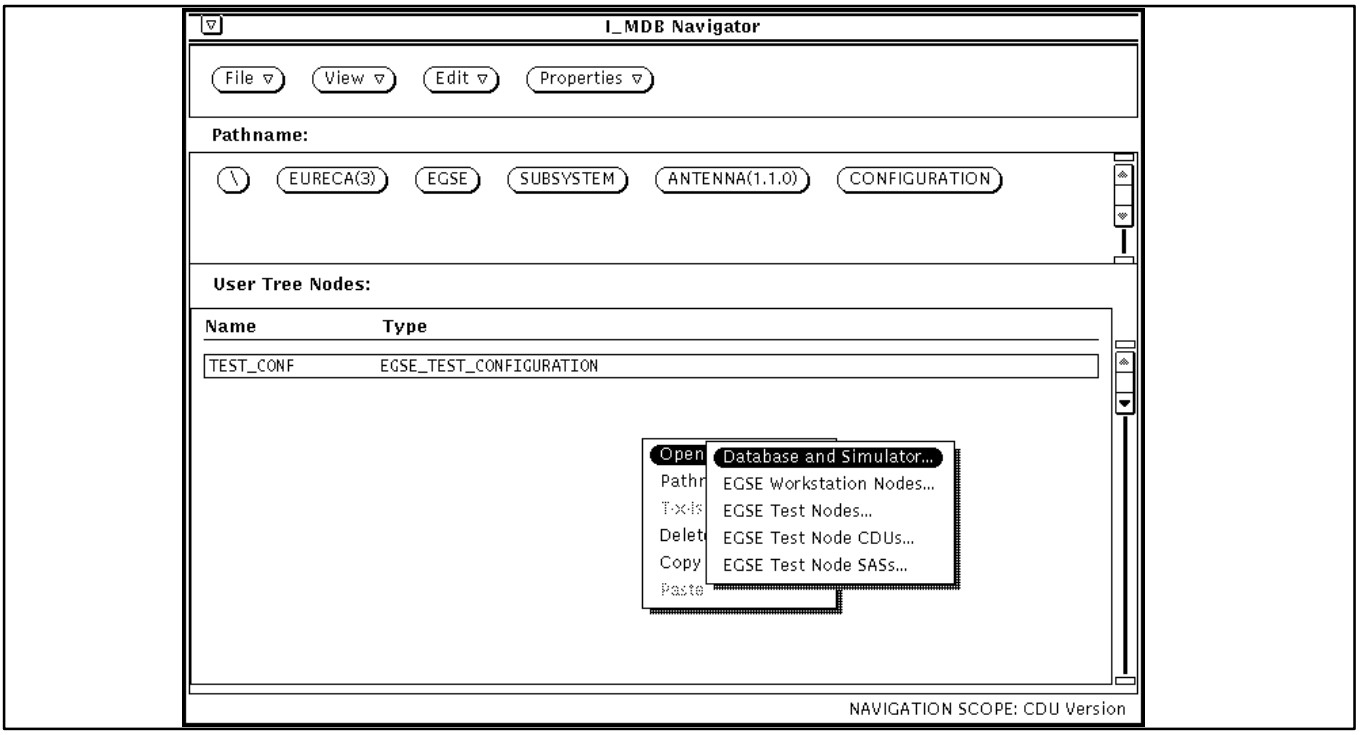


Figure 6–7 : *To Describe a Test Configuration all Information Must be Supplied Here*

Note that all the following procedures are needed to define a test configuration.
Instead of writing one large procedure the procedure has been split into several smaller ones to increase readability and comprehensibility.

Defining the Test Configuration – Database and Simulator Node

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- Create an end item of type EGSE_TEST_CONFIGURATION.
- Select **Open–>Database and Simulator** and an input window appears.
- Press the **Database Node ...:** button. An **Item Reference Help** window pops up.

to be continued ...

Copyright per DIN 34

- Press the **Select CDU versions...** button. A CDU Version Selections window appears displaying a list of available CDU version.
- Select the CDU version where the database node is located. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the database node and press the **Apply** button.
- The complete DB path including the database node end item is displayed in the corresponding input line (see Figure 6–8). Press the **apply** button again.
- *The definition of a simulation node is not mandatory.*
- Press the **Simulation Node ...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU versions...** button. A CDU Version Selections window appears displaying a list of available CDU version.
- Select the CDU version where the simulation node is located. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the simulation-node and press the **Apply** button.
- The complete DB path including the simulation node end item is displayed in the corresponding input line. Press the **apply** button again.

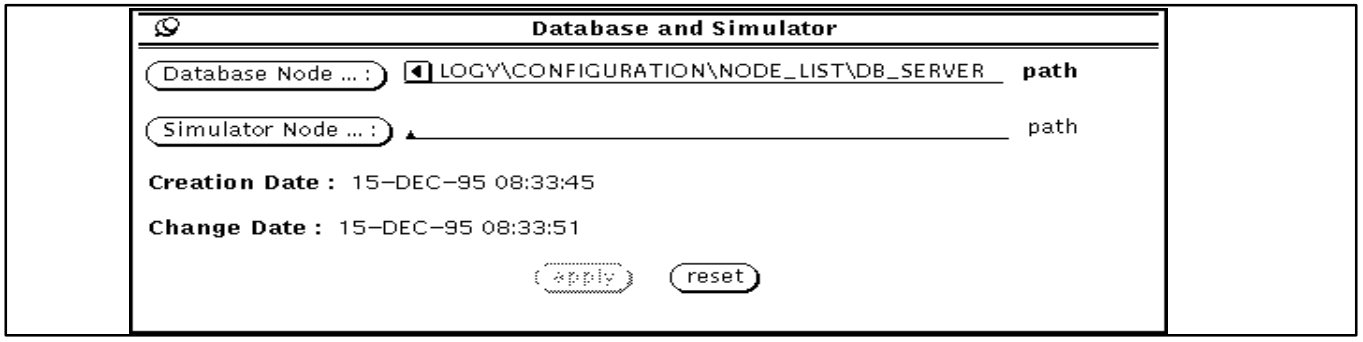


Figure 6–8 : *The Database and Simulator Nodes are Identified by their Pathname*

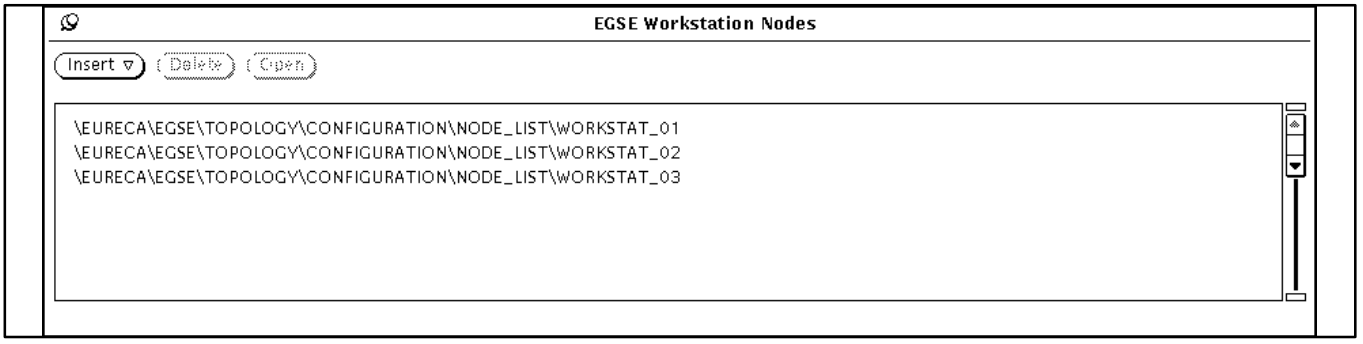


Figure 6–9 : *The Workstation Node List*

Copyright per DIN 34

Defining the Test Configuration – Workstation Nodes

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- *Repeat the following steps for each participating workstation node.*
- Select the test configuration end item.
- Select **Open–>EGSE Workstation Nodes...** and an **EGSE Workstation Nodes** list window pops up (see Figure 6–9).
- Press the **Insert:** button. An **EGSE Workstation Nodes** definition window pops up.
- Press the **Workstation Node...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where the workstation node is defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired workstation node and press the **Apply** button.
- Press the **Is Participating** button, select TRUE and press the **apply** button
- The complete DB path including the workstation node end item is displayed in the corresponding input line (see Figure 6–10). The **Is Participating** line shows TRUE. Press the **apply** button again.
- Close the window by clicking on the push pin.

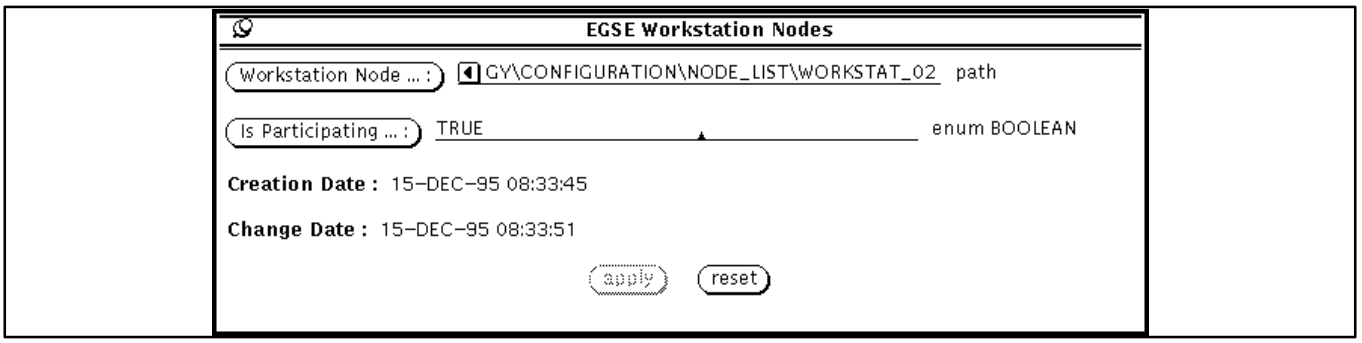


Figure 6–10 : *The Workstation Node is Participating*

- *In a test configuration there is at least one test node required.*

Copyright per DIN 34

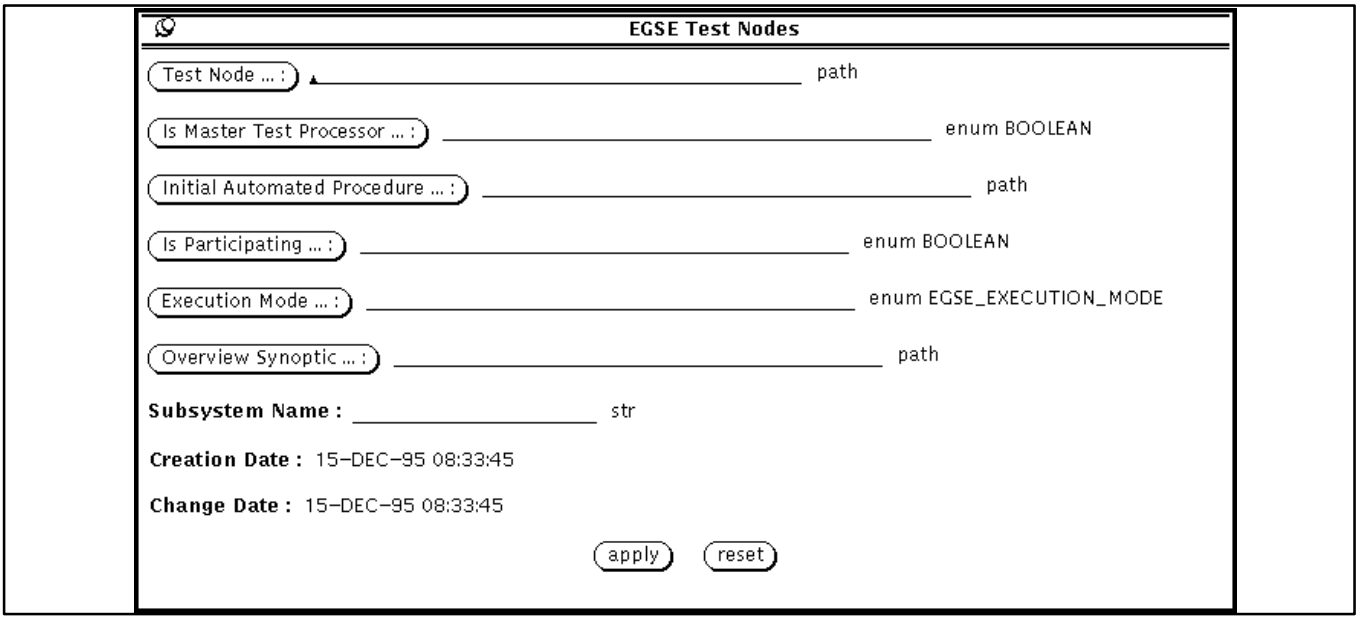


Figure 6–11 : *How To Create a Testnode in the Test Node Definition Window*

Defining the Test Configuration – Test Nodes

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- *Repeat the following steps for each participating test node.*
- Select the TEST_CONFIGURATION end item.
- Select **Open–>EGSE Test Nodes...** .
- Press the **Insert...** button and an **EGSE Test Nodes** definition window pops up (see Figure 6–11).
- Press the **Test Node...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where the test node is defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired test node and press the **Apply** button.
- *If there is only one test node in the configuration this must be the MTP (Master Test Processor) The definition of a MTP is mandatory for two reasons:*
  – setting the SMT (Simulated Mission Time) is done on the MTP only
  – there is a selection of HK (House Keeping) data only available from the MTP
- Press the **Is Master Test Processor...:** button. An **input** window pops up.
- Select TRUE and press the **apply** button.

to be continued .....

Copyright per DIN 34

- Perform the following steps for each additional participating test node.
- Press the **Is Master Test Processor...:** button. An **input** window pops up.
- Select FALSE and press the **apply** button.

The following input is not mandatory.
With the initial AP defaults can be set for a test node and an application can be started on the desired test node automatically.

- Press the **Initial Automated Procedure...:** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.
- Select the CDU version where APs are defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired AP and press the **Apply** button.
- Press the **Is Participating ... :** button, select TRUE and press the **apply** button

There are three execution modes: NORMAL, REPLAY and SIMULATION
In NORMAL mode the tests with the UUT are performed, in REPLAY mode the user will see the operations of a previously executed test, in SIMULATION mode all incoming data is generated within the test nodes itself (refer to section 3.2.2.6 for more information).

- the **Normal** test execution mode
  in this mode the nominal tests with the UUT are performed.
- the **Replay** mode
  in this mode the user will see the operations of a previously executed test. The replay submode shall be used to look to events anticipated by the operator which were not encountered during tests or to investigate situations before test deviations occurred. The data presentation will be in the same way as for normal mode, but all data that are generated to interfaces external to the checkout and test system are suppressed. The replay mode may be set up with different parameters w.r.t. to timing behaviour.
- the **Simulation** test execution mode
  in this mode all incoming data is generated within the test nodes itself. The simulation of this data is driven by predefined data tables. Simulation mode is normally used for database verification, i.e. test of synoptic displays etc.

- Press the **Execution Mode... :** button, select NORMAL and press the **apply** button
- The complete DB path including the workstation node end item is displayed in the corresponding input line (see Figure 6–12). The **Is Participating** line shows TRUE. Press the **apply** button again.

The following input is not mandatory.
With the overview synoptic a default screen layout can be determined.

- Press the **Overview Synoptic... :** button. An **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions.

to be continued .....

Copyright per DIN 34

- Select the CDU version where APs are defined. Then press the **Apply** button.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired synoptic display and press the **Apply** button.
- *The following input is not mandatory. It is made for documentation purposes.*
- Type the subsystem name in the **Subsystem Name:** line.
- Press the **Apply** button.
- Close the window by clicking on the push pin.

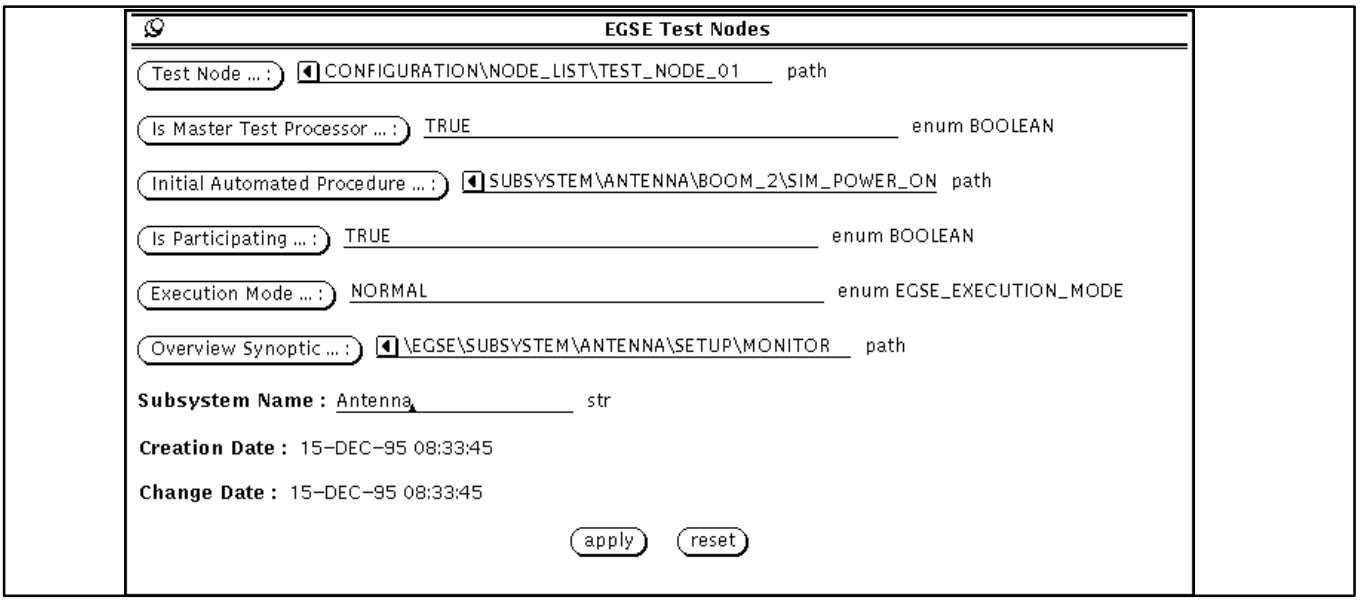


Figure 6–12 : *The Completed Test Nodes Window.*

It takes two steps to make data available on a test execution node:

* it has to be visible in the used CCU

* it has to be loaded to this execution node

Precondition for the execution of step two is the execution of the following procedure.
All CDUs which contain data needed for the test must be assigned to the test nodes. This includes CDUs with HK data, the configuration definition itself, displays, SASs, simulation models and SW variables, GDU description lists and APs, CCSDS Adu description and SW variables.

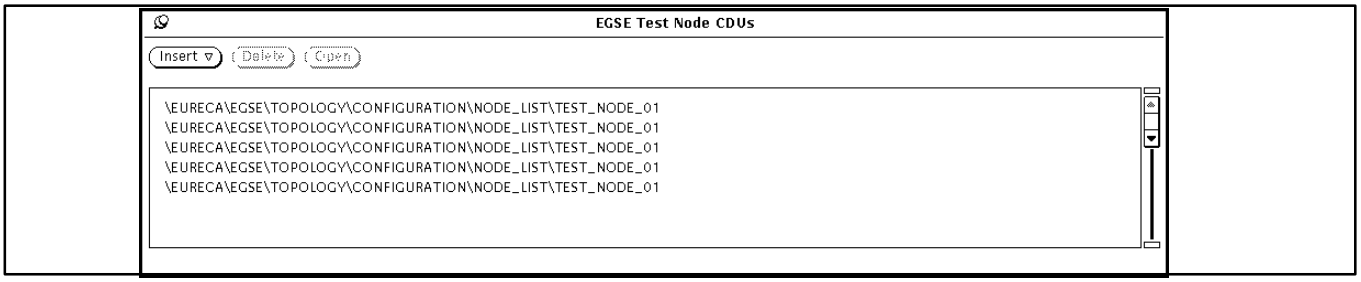


Figure 6–13 : *CDU List – Five CDUs are Assigned to Test Node TEST_NODE_01*

Copyright per DIN 34

Defining the Test Configuration – Test Node Items

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- Select the TEST_CONFIGURATION end item.
- Select **Open–>EGSE Test Node Items...** and an **EGSE Test Node Items** list window pops up (see Figure 6–13).
- *For each item (CDU, virtual node or enditem) which contains things connected to the test node a reference has to be set.*
- Press the **Insert:** button. An **EGSE Test Node Items** definition window pops up.
- Press the **Test Node...:** button. An **Item Reference Help** window pops up.
- The **Item Reference Help** window shows now a list of all entries in that CDU. Select the desired test node and press the **Apply** button.
- *Note that it is possible to select a CDU which includes other CDUs.*
- Press the **Loaded Item...** button. An **Item Reference Help** window pops up.
- The **Item Reference Help** window shows now the selected items. Press the **Apply** button.
- The complete DB path including the test node end item is displayed in the corresponding input line (see NO TAG). The **Loaded item...** line shows the DB path of the selected CDU.
- Repeat the last five steps for all items needed on that test node.
- Close the window by clicking on the push pin.

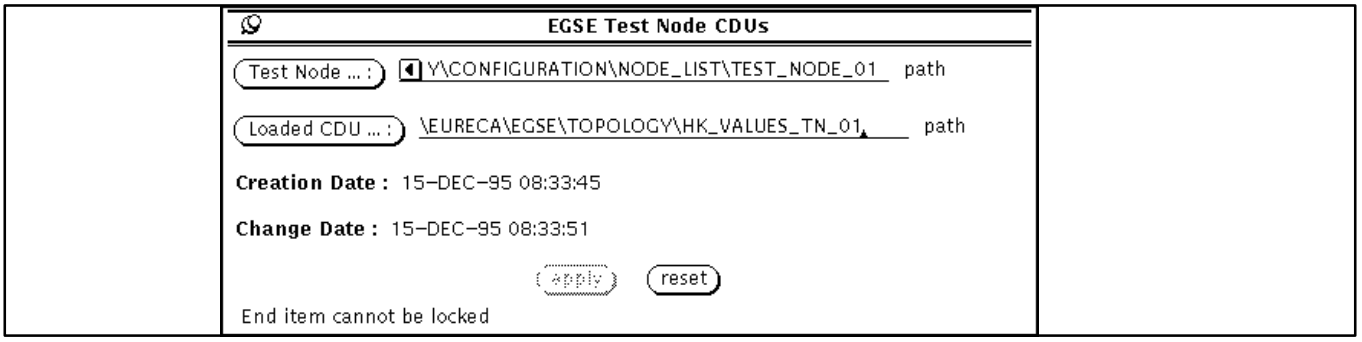


Figure 6–14 : *The House Keeping Values Defined for TEST_NODE_01 are Assigned to the Test Node*

Copyright per DIN 34

Defining the Test Configuration – Test Node SASs

- If not already done, create a CDU (domain: EGSE) for the test configuration definitions.
- For each test configuration repeat the following steps.
- If not already done, create an end item of type EGSE_TEST_CONFIGURATION.
- *Repeat this steps for all test nodes running SASs.*
- Select **Open–>Test Node SASs.**
- Press the **Insert:** button. An **EGSE Test Node SASs** definition window pops up.
- Press the **Test Node...:** button and select the desired test node as already described.
- Press the **SAS...** button.
- The **Item Reference Help** window pops up.
- Press the **Select CDU version...** button. A **CDU Version Selection** window appears displaying a list of available CDU versions. Select a CDU and press the **Apply** button.
- The **Item Reference Help** window shows now a list of all SASa in that CDU. Select the desired SAS and press the **Apply** button.
- The complete DB paths including the test node end item and the SAS end item are displayed in the corresponding input lines (see Figure 6–10).
- *Note that the SAS has to be assigned to the test node in the SYSTEM_TOPOLOGY_TABLE. The database entry repeats the assignment already made in the table.*

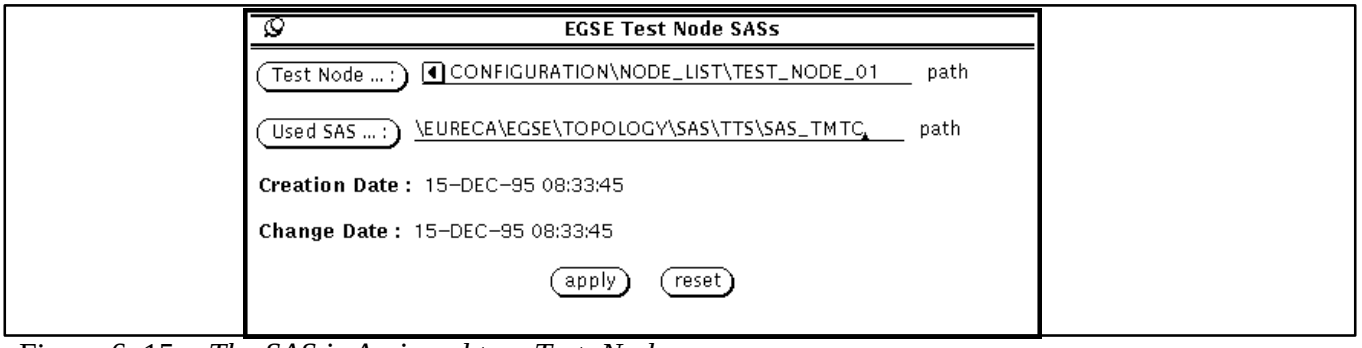


Figure 6–15 : *The SAS is Assigned to a Test Node*

6.1.4 Defining House Keeping (HK) Values

In section 6.1.12 a list of TES housekeeping data is provided.

There are no mandatory house keeping variables. Software variables are only needed for those TES housekeeping data that shall be available for APs and SASs or when values shall be displayed in synoptics.

We recommend to separate the housekeeping data into different CDUs, one for the Master Test Processor (MTP) house keeping values and one for each test node. The MTP CDU contains those HK values which are used for global overall monitoring.

Note that this variables should be defined just once per system tree version and being referenced in the different CDUs.

Copyright per DIN 34

A software variable is to be created in the data base with a reference to the HK value. The housekeeping identifier is the number which can be found in the list. The data type must comply with the data type listed in section 6.1.12.

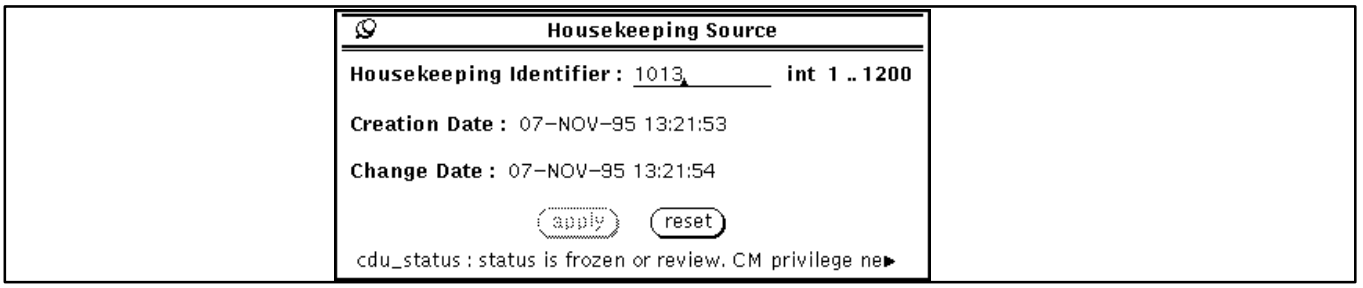

Housekeeping Source
Housekeeping Identifier : 1013
int 1 .. 1200
Creation Date : 07-NOV-95 13:21:53
Change Date : 07-NOV-95 13:21:54
apply
reset
cdu_status : status is frozen or review. CM privilege ne

Figure 6–16 : *The Housekeeping Identifier is a Number from the List of HK Data*

6.1.5 Defining UCL Libraries

6.1.5.1 Defining UCL System Libraries for VICOS / TES

The baselined version of the CGS system libraries can be found in your installation:
Under $CGS_HOME/lib/ucl the following files exist:

ground_common_.ucl	This library contains constants and type specifications used within all other system libraries
atv_library_.ucl	This library contains operations to set/get sequence counters, to get CCSDS_unsegmented time and to issue time tagged TC's
grd_conversion_.ucl	Defines Procedures to convert items within the GROUND system
ground_library_.ucl	This library is needed always to control VICOS/TES (all projects)
cpl_library_.ucl	contains the library used for CPL (Columbus)
fwdu_library_.ucl	contains the library needed for the FWDU (Columbus)
ground_commands_to_onboard_.ucl	contains the library needed for APM commands (Columbus)
ground_values_.ucl	a library four ground operations on values (measurements, sw variables and derived values)
packet_library_.ucl	Defines Procedures to maintain TM packet buffers and Trigger APs
raw_data_library_.ucl	Defines Procedures to retrieve data from raw TM packets
tc_construction_.ucl	Defines Procedures to construct Telecommands

Your configuration has to contain at least one end item containing the UCL specification of the "Ground Library". Create in your CCU/CDU an end item of type UCL_SYSTEM LIBRARY and open it with the CLS Editor. Load the contents of the file "ground_library_.ucl" into the editor. Compile the library and store it.

The other libraries are optional. If you need them, depends on your configuration.

Copyright per DIN 34

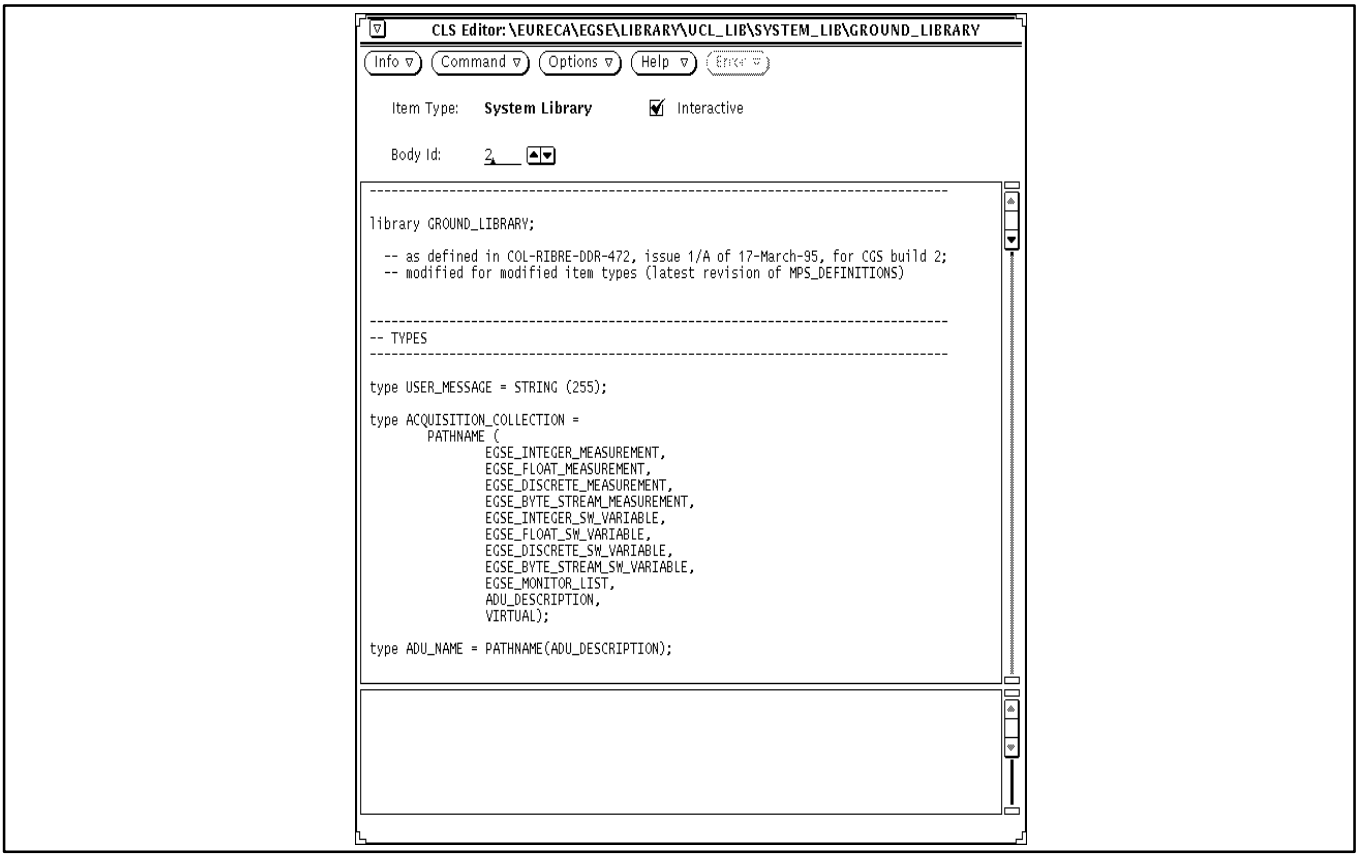


Figure 6–17 : *Ground Library Loaded into CLS Editor*

Figure 6–17 shows how the CLS Editor has to be set up before compiling the Ground Library. The ”Body Id” has to be set up according to the library definition in TES. For the Ground Library, the body id is 2.

6.1.5.2 Defining UCL User Libraries

The baselined version of the CGS some user libraries can be found in your installation:
Under $CGS_HOME/config the following files exist:

file_io_lib_.ucl	FILE_IO_LIB User Library Specification. This library provides acces to UNIX files from UCL APs
file_io_lib.ucl	Body of the FILE_IO_LIB library

To install the library , create in your CCU/CDU an end item of type UCL_USER LIBRARY and open it with the CLS Editor. Load the contents of the file ”file_io_lib_.ucl” into the editor. Switch to ”BODY” and load the contents of the file ”file_io_lib.ucl” into the editor. Compile the library and store it.

Further libraries can be developped by defineing new enditems of type UCL_USER LIBRARY and creating your own UCL code.

Copyright per DIN 34

6.1.6 Deleted

6.1.7 Defining the User Specific Configuration

In the I_MDB end items for all measurements, commands, ADUs, APs or Synoptic Displays used during the test have to be created and the current allocation must be made. Note that APs have to be compiled before using. For this purpose you need to establish a development CCU to make the right environment visible.

6.1.8 Defining SASs

Before starting the implementation of an SAS, a proper environment set–up has to be ensured in terms of setting–up

* the UNIX environment
* the compiler system environment

For these two steps contact your responsible system administrator.

For each SAS an end item of type EGSE_SW , which defines the SAS, has to be created.
The application name for SASs have to correspond exactly to the short names defined for the SAS in the MDB. For a detailed procedure how to create SASs refer to section 7.2.

For version 6.1 of CGS (and later) it is not necessary to define SAS ’s in the System Topology Table.

6.1.9 Building a CCU

It takes two steps to make the information available on a test execution node:

* it has to be visible in the used CCU
* it has to be loaded to this execution node

The precondition for step 2 is the execution of procedure ”**Defining the test configuration – Test node items**” described in section 6.1.3.2 .
How to create a CCU is described in detail in section 5.1.2.3.
Create references to all CDUs that are necessary to define the test configuration (see Figure 6–18)

Copyright per DIN 34

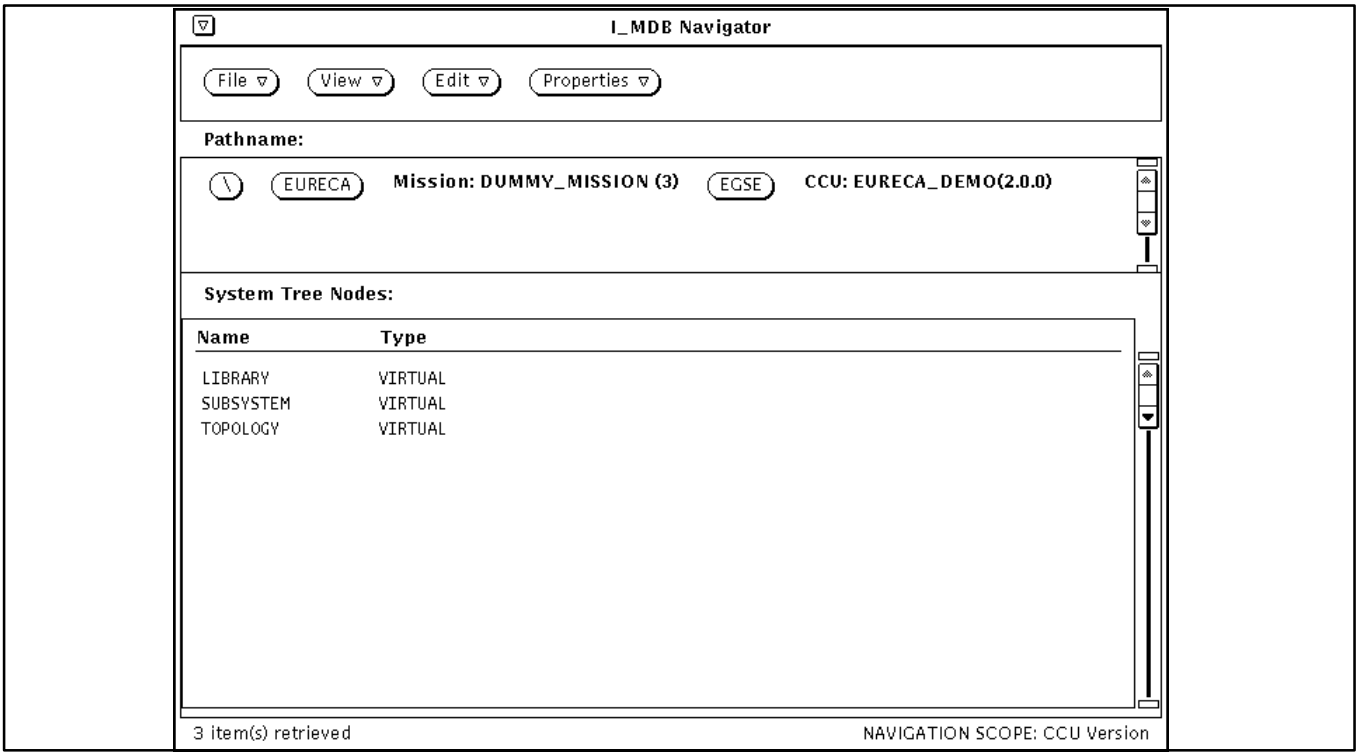


Figure 6–18 : *The CCU EURECA_DEMO References all CDUs which Contents is Needed for the Test*

6.1.10 Performing Consistency Checks

Execute the consistency checker on the new/changed CCU and eliminate the marked problems. Specially all reference errors have to be solved before generating the Load Scoe file.

Copyright per DIN 34

6.1.11 Generating the Scoe Load File

Note that the CCU has to be selected before the following steps can be performed.

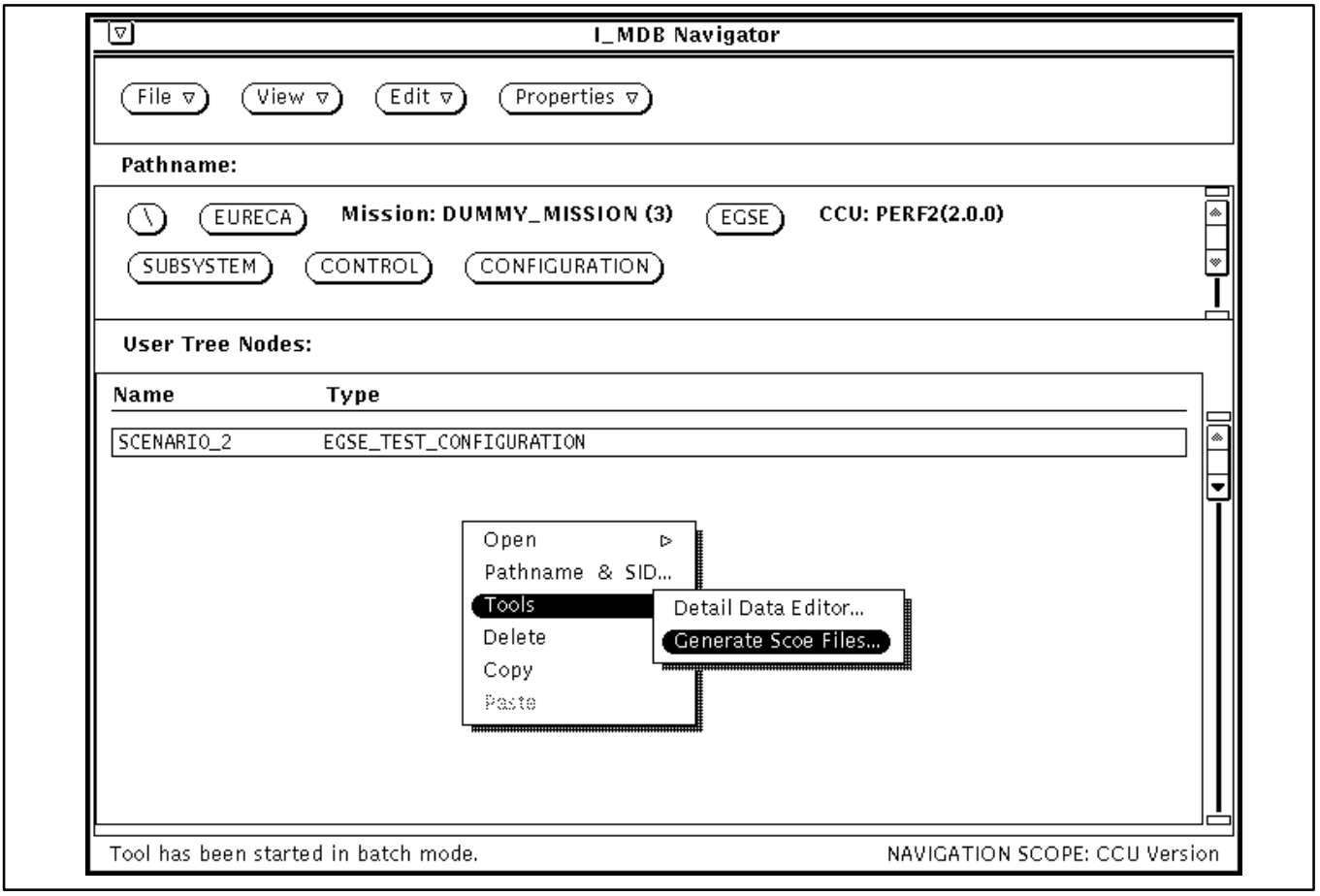


Figure 6–19 : *I_MDB Navigator Provides the Option to Start the Scoe File Generation Process*

Generate the Scoe file for each EGSE Test Configuration you may want to select via TSCV lateron in this CCU by selecting **Tools–>Generate Scoe Files** in the corresponding EGSE_TEST_CONFIGURATION end item menu for this node (Figure 6–19). Look in the Console window for generation messages:

Copyright per DIN 34

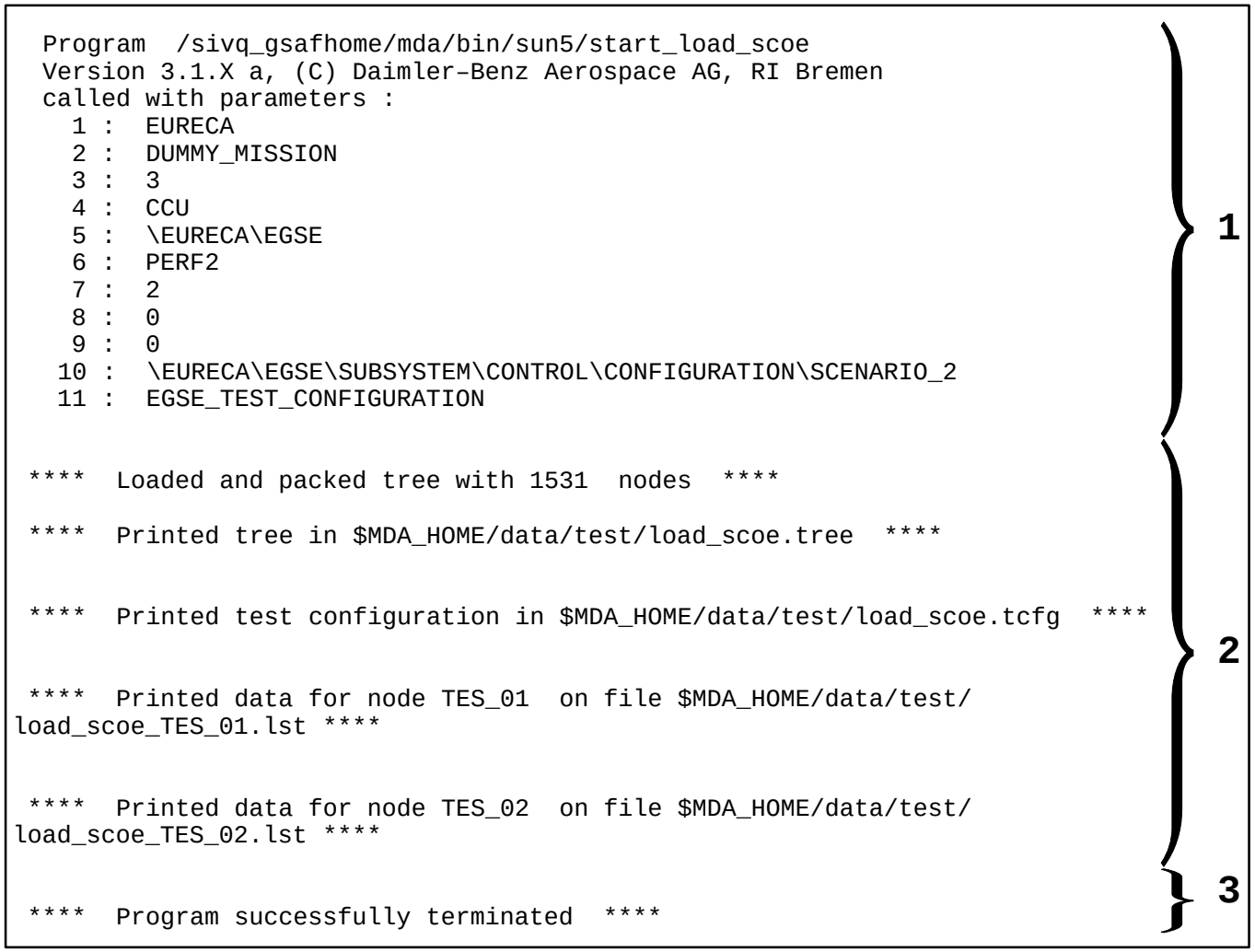

Program /sivq_gsafhome/mda/bin/sun5/start_load_scoe
Version 3.1.X a, (C) Daimler-Benz Aerospace AG, RI Bremen
called with parameters :
1 : EURECA
2 : DUMMY_MISSION
3 : 3
4 : CCU
5 : \EURECA\EGSE
6 : PERF2
7 : 2
8 : 0
9 : 0
10 : \EURECA\EGSE\SUBSYSTEM\CONTROL\CONFIGURATION\SCENARIO_2
11 : EGSE_TEST_CONFIGURATION
1
**** Loaded and packed tree with 1531 nodes ****
**** Printed tree in $MDA_HOME/data/test/load_scoe.tree ****
**** Printed test configuration in $MDA_HOME/data/test/load_scoe.tcfg ****
**** Printed data for node TES_01 on file $MDA_HOME/data/test/load_scoe_TES_01.lst ****
**** Printed data for node TES_02 on file $MDA_HOME/data/test/load_scoe_TES_02.lst ****
2
**** Program successfully terminated ****
3

Figure 6–20 : *Output of Start_load_scoe in Console Window*

The first block of messages (Figure 6–20, section 1) describes the version of the ”Start Load Scoe” program and the parameters passed to it. Then the following output (Figure 6–20, section 2) lists the names of the files generated during the Scoe file generation process. These files are generated mainly for debugging purposes. In the example above these are:

`$MDA_HOME/data/test/load_scoe.tree:` Contains the name tree data as written to the Scoe file.

`$MDA_HOME/data/test/load_scoe.tcfg:` Contains the test configuration data as written to the Scoe file.

`$MDA_HOME/data/test/load_scoe_TES_01.lst:` Contains the listing for the data for the first test node (in this case TES_01).

`$MDA_HOME/data/test/load_scoe_TES_02.lst:` Contains the listing for the data for the second test node (in this case TES_02).

- *The generation of the Scoe file was only successful when the message ”Program successfully terminated” is displayed (Figure 6–20, section 3).*

Copyright per DIN 34

6.1.12 List of Available TES HK DATA

Id	Name	Purpose	Type	Initial Value
1	”Init_Path”	”Pathname of EGSE_TEST_CONFIGURATION item, used for LOAD_SCOE”	STRING_TYPE	””
2	”TES_Mode”	”Current Mode (NONE, NORMAL, REPLAY, SIMULATE)”	STATE_CODE_TYPE	$NONE
3	”TN_Status”	”TN Status (IDLE, RUNNING, SUSPEND, ERROR)”	STATE_CODE_TYPE	$IDLE
4	”MTP_Mode”	”The value for the MTP mode (SCOE, MTP)”	STATE_CODE_TYPE	$SCOE
5	”CCU internal version”	MDA internal identifier of the CCU version used	INTEGER_TYPE	0
6	”Replay_Speed”	”The replay speed”	INTEGER_TYPE	100
7	”Archive_Time”	”The cycle time (min) an archive file will be closed”	INTEGER_TYPE	0
8	”Archive_State”	”The current state of archiving (DISABLED, ENABLED)”	STATE_CODE_TYPE	$DISABLED
10	”Local_Time”	”Actual Local Clock Value”	STRING_TYPE	””
11	”SMT”	”Actual Simulated Mission Time”	STRING_TYPE	””
12	”RLT”	”Recorded Local Time as read from archived data”	STRING_TYPE	””
20	”Active_APs”	”Number of started (but not terminated) APs”	INTEGER_TYPE	0
21	”Susp._APs”	”Number of APs explicitely suspended”	INTEGER_TYPE	0
30	”RPL_Begin_Time”	”Replay begin time selected during init”	STRING_TYPE	””
31	”RPL_End_Time”	”Replay end time selected during init”	STRING_TYPE	””
90	”Free_Disk”	”The free disk space on local disk ($TN_HOME) in kBytes”	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
100	”SAS1_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
101	”SAS1_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
102	”SAS1_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
103	”SAS1_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
104	”SAS1_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
105	”SAS2_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
106	”SAS2_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
107	”SAS2_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
108	”SAS2_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
109	”SAS2_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
...				
194	”SAS19_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
195	”SAS20_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
196	”SAS20_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
197	”SAS20_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
198	”SAS20_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
199	”SAS20_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
200	”AP_1_Name”	”Pathname of AP”	STRING_TYPE	””
201	”AP_1_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
202	”AP_1_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
203	”AP_1_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
204	”AP_1_ID”	”The AP identifier”	INTEGER_TYPE	0
205	”AP_2_Name”	”Pathname of AP”	STRING_TYPE	””
206	”AP_2_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
207	”AP_2_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
208	”AP_2_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
209	”AP_2_ID”	”The AP identifier”	INTEGER_TYPE	0
210	”AP_3_Name”	”Pathname of AP”	STRING_TYPE	””
...				
294	”AP_19_ID”	”The AP identifier”	INTEGER_TYPE	0
295	”AP_20_Name”	”Pathname of AP”	STRING_TYPE	””
296	”AP_20_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
297	”AP_20_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
298	”AP_20_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
299	”AP_20_ID”	”The AP identifier”	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
301	”Nb._Enditems”	”Number of enditems that can be monitored”	INTEGER_TYPE	0
302	”Nb._Enabled”	”Number of enditems enabled for monitoring”	INTEGER_TYPE	0
303	”Nb._Digital”	”No. of digital enditems that can be monitored”	INTEGER_TYPE	0
304	”Nb._Grp_Discr.”	”No. of group discrete enditems that can be monitored”	INTEGER_TYPE	0
305	”Nb._Analog”	”No. of analog enditems that can be monitored”	INTEGER_TYPE	0
306	”Nb._bytestream”	”No. of bytestream enditems that can be monitored”	INTEGER_TYPE	0
310	”Nb_Soft”	OOL Number of enditems currently out of soft limit(1)	INTEGER_TYPE	0
311	”Nb_Hard”	OOL Number of enditems currently out of hard limit (1)	INTEGER_TYPE	0
312	”Nb._Soft_exceptions ”	Number of soft limit violations since last START(1)	INTEGER_TYPE	0
313	”Nb._Hard_exceptions ”	Number of hard limit violations since last START(1)	INTEGER_TYPE	0
320	”Nb. Acquired”	Number of measurements currently acquired	INTEGER_TYPE	0
321	”Nb. EVL”	Number of acquired measurements or SW variables or derived values with EVL enabled	INTEGER_TYPE	0
322	”Nb. Measurements”	Number of meaurements defined/loaded to the test node	INTEGER_TYPE	0
323	”Nb. SW_Variables”	Number of software variables defined/loaded to the test node	INTEGER_TYPE	0
324	”Nb. Derived_Values	Number of derived values defined/loaded to the test node	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
400	”Nb._stimuli”	”Number of GDUs sent out since last START”	INTEGER_TYPE	0
401	”Stimuli_Errors”	”Number of GDUs with errors in SAS/FEE since last START”	INTEGER_TYPE	0
402	”Wrong_Stimulus”	”Pathname of last stimulus that resulted in an error”	STRING_TYPE	””
404	”Destination”	”Destination of last erroneous simulus (SAS name)”	STRING_TYPE	””
410	”Nb._of_digital_GDUs”	”Number of digital output GDUs (loaded from MDB)”	INTEGER_TYPE	0
411	”Nb._of_analog_GDUs”	”Number of analog output GDUs (loaded from MDB)”	INTEGER_TYPE	0
412	”Nb._of_TC_GDUs”	”Number of GDUs with CCSDS TCs (loaded from MDB)”	INTEGER_TYPE	0
413	”Nb._of_GDUs”	”Total number of GDUs (loaded from MDB)”	INTEGER_TYPE	0
414	”Nb._of_bin.pack.GDUs”	”Number of binary packet GDUs (loaded from MDB)”	INTEGER_TYPE	0
415	”Nb.GDU_Verif_Succ”	Number of GDUs sent with successful verification	INTEGER_TYPE	0
416	”Nb.GDU_Verif_Fail”	Number of GDUs sent with failed verification	INTEGER_TYPE	0
420	”Nb._of_SW_cmd._sent”	”Number of SW commands sent”	INTEGER_TYPE	0
421	”Nb._of_rtn._packets”	”Number of return packets received for SW commands”	INTEGER_TYPE	0

Table 6–1 : *Housekeeping Data Description*

REMARKS:

1. An end item is only considered ”out–of–limits” if the limit is violated and the error count has been reached with respect to these housekeeping values. In the transition period (i.e. the limit is violated but the error count has not yet been reached) the housekeeping values will not change

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
500	”AP_21_Name”	”Pathname of AP”	STRING_TYPE	””
501	”AP_21_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
502	”AP_21_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
503	”AP_21_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
504	”AP_21_ID”	”The AP identifier”	INTEGER_TYPE	0
...	...	...	...	...
595	”AP_40_Name”	”Pathname of AP”	STRING_TYPE	””
596	”AP_40_Status”	”Status (NOT_RUN, INITIAL, RUNNING, SUSPEND, TERMINAT)”	STATE_CODE_TYPE	$NOT_RUN
597	”AP_40_Stmt.”	”Current UCL Statement of AP”	INTEGER_TYPE	0
598	”AP_40_HCI_ID”	”The ID of the HCI that started this AP or its parent”	STRING_TYPE	””
599	”AP_40_ID”	”The AP identifier”	INTEGER_TYPE	0
600	”Served_SAS”	Number of connected special application software (SAS)	INTEGER_TYPE	0
601	”Served_WSs”	Number of connected workstations (HCI)	INTEGER_TYPE	0
602	”Served_CIS”	Number of connected CGS interface server (CIS)	INTEGER_TYPE	0
700	”Nb.Cond_Enditems”	Number of Enditems carrying Conditions	INTEGER_TYPE	0
701	”Nb.Conditions”	Number of Conditions defined	INTEGER_TYPE	0
702	”Nb.TriggeredCond”	Number of actions triggered by conditions since last start	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
800	”SAS21_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
801	”SAS21_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
802	”SAS21_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
803	”SAS21_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
804	”SAS21_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
805	”SAS22_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
806	”SAS22_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
807	”SAS22_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
808	”SAS22_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
809	”SAS22_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
...				
894	”SAS39_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0
895	”SAS40_Name”	”Name of Appl. Progr. connected to TES”	STRING_TYPE	””
896	”SAS40_Service”	”SAS service (NONE, ADU_SERV, GDU_SERV, ADU_GDU)”	STATE_CODE_TYPE	$NONE
897	”SAS40_Errors”	”The number of error messages sent by this SAS”	INTEGER_TYPE	0
898	”SAS40_Last_Err”	”The last error message sent by this SAS”	STRING_TYPE	””
899	”SAS40_Link_ID”	”The Link ID for the SAS”	INTEGER_TYPE	0

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
900	”Time_of_Exception”	Time of last Monitoring exception (in seconds since midnight)	FLOAT_TYPE	0.0
901	”Time_of_Start_AP”	Time of last Start AP (in seconds since midnight)	FLOAT_TYPE	0.0
902	”Time_of_Resume_AP”	Time of last Resume AP (in seconds since midnight)	FLOAT_TYPE	0.0
903	”Nb_Processed_ADUs”	number of processed ADUs on this testnode (1)	INTEGER_TYPE	0
904	”Nb_Discarded_ADUs”	number of discarded ADUs on this testnode (1)	INTGER_TYPE	0

Table 6–2 : *Housekeeping Data Description (continued)*

REMARKS:

1. This HK values are disable by default, due to some performance reasons. To enable this HK values, change in the cgs_configuration.xml the parameter TES.ADU.WRITE_HK_VALUE (unhide and change) and restart the vicos_tes process via TSCV programm.

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
1001	”PR1_Status”	”The status of printer1 (DISABLED, ENABLED, OTHERS)”	STATE_CODE_TYPE	$DISABLED
1002	”PR2_Status”	”The status of printer2 (DISABLED, ENABLED, OTHERS)”	STATE_CODE_TYPE	$DISABLED
1003	”PQ1_Status”	”Status (OFF, READY, PRINTING, NO_PAPER, NO_TONER)”	STATE_CODE_TYPE	$OFF
1004	”PQ2_Status”	”Status (OFF, READY, PRINTING, NO_PAPER, NO_TONER)”	STATE_CODE_TYPE	$OFF
1005	”PQ1_Jobs”	”The number of jobs in print queue 1”	INTEGER_TYPE	0
1006	”PQ2_Jobs”	”The number of jobs in print queue 2”	INTEGER_TYPE	0
1011	”MD_Free_Spac”	”The free disc space on TRDB disk ($VICOS_CEN_DBS_HOME) in kByte”	INTEGER_TYPE	0
1012	”FA_1_Dev_Status”	”The device status (OK, NOT_OK) of the long term storage medium”	STATE_CODE_TYPE	$OK
1020	”TRDB_Over-all_status”	”The overal status of the test result data base (OK, NOT_OK)”	STATE_CODE_TYPE	$OK
1021	”Cmdh_Over-all_status”	”The overal status of the command history (OK, NOT_OK)”	STATE_CODE_TYPE	$OK
1050	”Session_Name”	”The name of the current test execution session”	STRING_TYPE	””
1051	”TRDB_Eval_U”	”The number of evaluation users connected to TRDB”	INTEGER_TYPE	0
1060	”TRDB_Event_Usage”	”The percentage of event table space used”	INTEGER_TYPE	0
1061	”TRDB_MA_Usage”	”The percentage of master archive table space used”	INTEGER_TYPE	0
1062	”TRDB_Misc_Usage”	”The percentage of miscel-laneous table space used”	INTEGER_TYPE	0
1100	”T_SYNC”	”The local clock is syn-chronized with NTP (FALSE, TRUE)”	STATE_CODE_TYPE	$FALSE

Copyright per DIN 34

Id	Name	Purpose	Type	Initial Value
1101	”MTU_PRESENT”	”Has an external master time unit (FALSE, TRUE)”	STATE_CODE_TYPE	$FALSE
1102	”SMT_STATUS”	”The status of the SMT (NOT_INIT, STOPPED, RUNNING)”	STATE_CODE_TYPE	$NOT_INIT
1103	”SYSTEM_LT_SYNC_ STATUS”	”The system synchronisation status of the LT (only on Master valid)” Alternative — meaning UNDEFINE — no valid statement possible TRUE — system is synchronized FALSE — system is not synchronized ConnLost — one (or more) client don’t answer NoUpdate — the local information will not be updated NoMaster — the current node is not time domain master	STATE_CODE_TYPE	$UNDEFINE
1130 .. 1200	User defined HK values	The user has the possibility to define discrete HK values in the cgs_configuration.xml (Housekeeping), which can be set by special application software (SAS) via TES_API procedure call.	STATE_CODE_TYPE	user defined

Table 6–3 : *Housekeeping Data Description (continued)*

Copyright per DIN 34

6.2 Preparing Special Application Software

6.2.1 Introduction

6.2.1.1 SAS Concept in CGS

This chapter covers two topics:

- how to implement an SAS and
- how to command an TES SAS in the CGS environment

Main focus will be put on guide-lines how special application software (SAS) shall be written in the CGS context, the second point will be mentioned briefly.

For CGS, SAS can be developed for different purposes:

- TES SAS
  - to interface between frontends and the TES process running on test nodes
  - to process data on top of the standard CGS facilities during ongoing tests
- TEV SAS
  - to evaluate data on top of the standard CGS evalution features

6.2.1.1.1 TES SAS

TES SAS constitutes the data interface between CGS and the front end equipment (FEE) directly interfacing to the test article, referred to as ’unit under test’ (UUT) here. Also, TES SAS adds special data processing features to a CGS system which the user needs in order to run the tests.

The general concept in CGS is that the interface to the unit under test (UUT) is established by special front end equipment (FEE) providing the direct interface on hardware level plus special application software (SAS) providing the ’glue’ between the FFE and CGS itself. Through this interface, the data from the UUT are acquired in raw data format and stimuli are sent to it.

Two different acquisition processes are supported by the SAS:

- data acquisition from the UUT
  i.e. the SAS acquires user selected data from the UUT and commands the UUT via the FEE at the speed of the employed hardware
- data delivery to CGS
  i.e. SAS delivers the acquired UUT data packetized in ADUs to CGS and receives commands contained in GDUs from CGS, both according to user specifications laid down in the related ADU and GDU descriptions

In CGS, the product TES provides this interface to SAS in the form of an application program interface (API). The TES_API is in fact a collection of Ada packages which have to be included into the SAS code during compilation. Thus, a high level interface on procedural level is constituted facilitating easy use and reliable data exchange.

Copyright per DIN 34

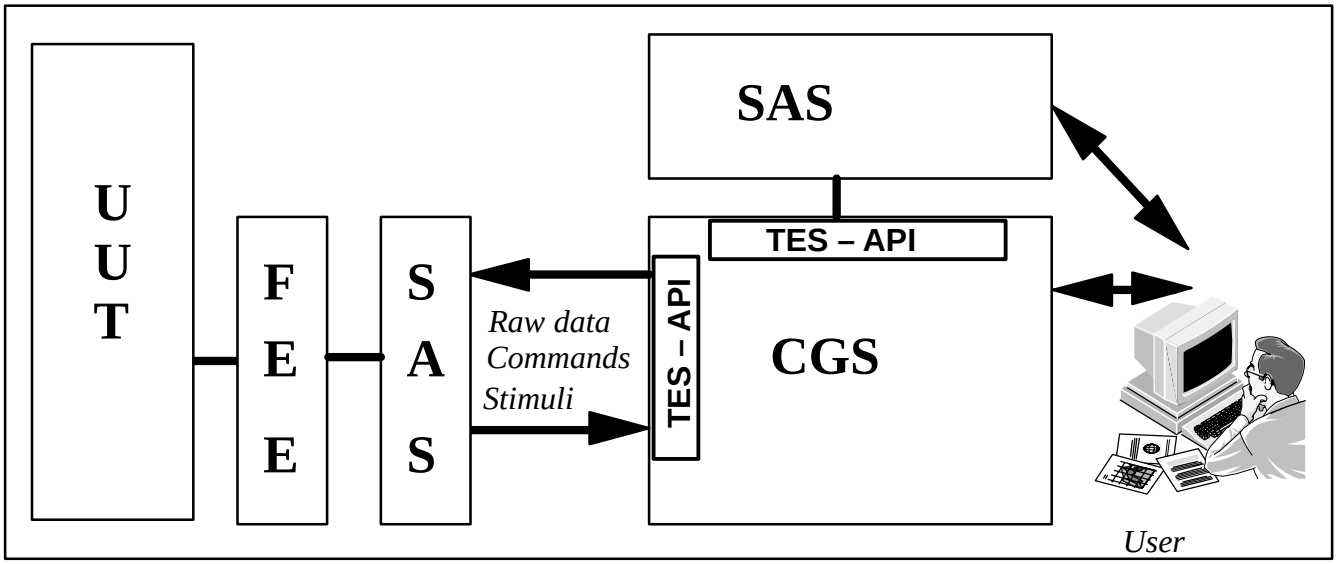


Figure 6–21 : *The Overall Concept of TES SAS together with CGS*

In addition, SAS is foreseen in all those areas where the user needs very special data processing features not provided by CGS. This can be special data visualisation, storing or calculations.

SAS are separate operating system processes running under CGS control and communicating with CGS via standardised, internal mechanisms. The SAS programmer does not have to care for the low level communication with CGS because this is hidden inside the API.

6.2.1.1.2 TEV SAS

TEV SAS constitutes the Softare on top of CGS evaluation facilities. It adds special data processing features to a CGS system which the user needs in order to evaluate the data stored during tests.

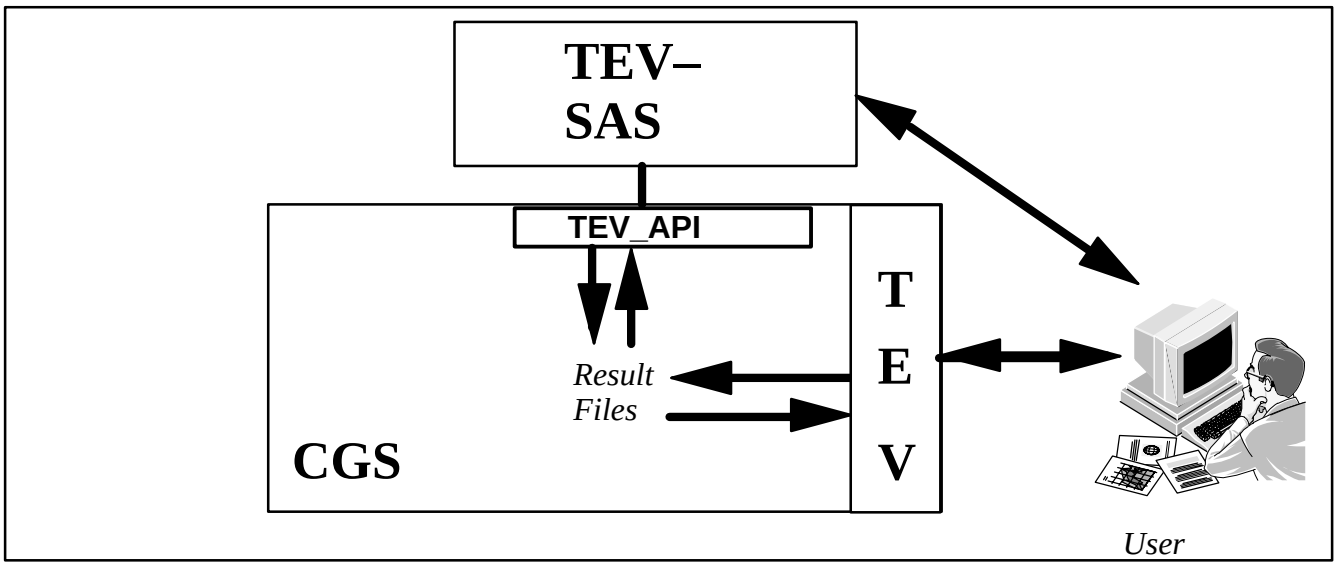


Figure 6–22 : *The Overall Concept of TEV SAS together with CGS*

Copyright per DIN 34

6.2.1.2 SAS Implementation Rules

Special Application Software (SAS) are programs written in Ada which have to run on test nodes. For the development of an SAS the Ada Compiler System Environment is used.

Part of the CGS delivery is the directory $GSAF_HOME/cgs/lib/cgs_api which contains subdirectories for each architecture supported (sun5/hp_ux8). Each of these contains the following

- the subdirectory EXT_ICD.<architecture>.source_tree containing the Ada and C source files of the API
- the file EXT_ICD.<architecture>, a list of the Ada source files describing the API (Ada Specifications)
- the file EXT_ICD.<architecture>.closure, a list of the Ada source files implementing the whole API
- the file makefile.<architecture> compiling the Ada source files and
- a README file explaining how to use CGS_API
- the bin directory containing link scripts for each type of SAS. These script are to be called for a sas giving the name of the SAS main program as first parameter (e.g. link_tes_sas_sun SAS_1)
  - link_dbs_sas
    script to link a DBS SAS on Sun
  - link_tes_sas_hp
    script to link a TES SAS on HP
  - link_tes_sas_sun
    script to link a TES SAS on Sun
  - link_tev_sas
    script to link a TEV SAS on Sun
  - linkada
    general script to be called by link_tev_sas (links any program by inlcuding the Oracle Interface Libraries)

SAS programs that use the CGS_API should be developed in a separate Ada library. The CGS_API Ada library must be found in the parent library tree.

The next figure below illustrates the set-up of Ada libraries necessary to implement SAS. On the root level there has to be the standard library provided by Alsys. The TES_API (Test Execution Software Application Programming Interface) provides the interface to CGS for SASs. Therefore this library has to be the parent library for all SASs. If several SAS are developed in parallel and contain common parts it is recommended to create an additional SAS common library.

Copyright per DIN 34

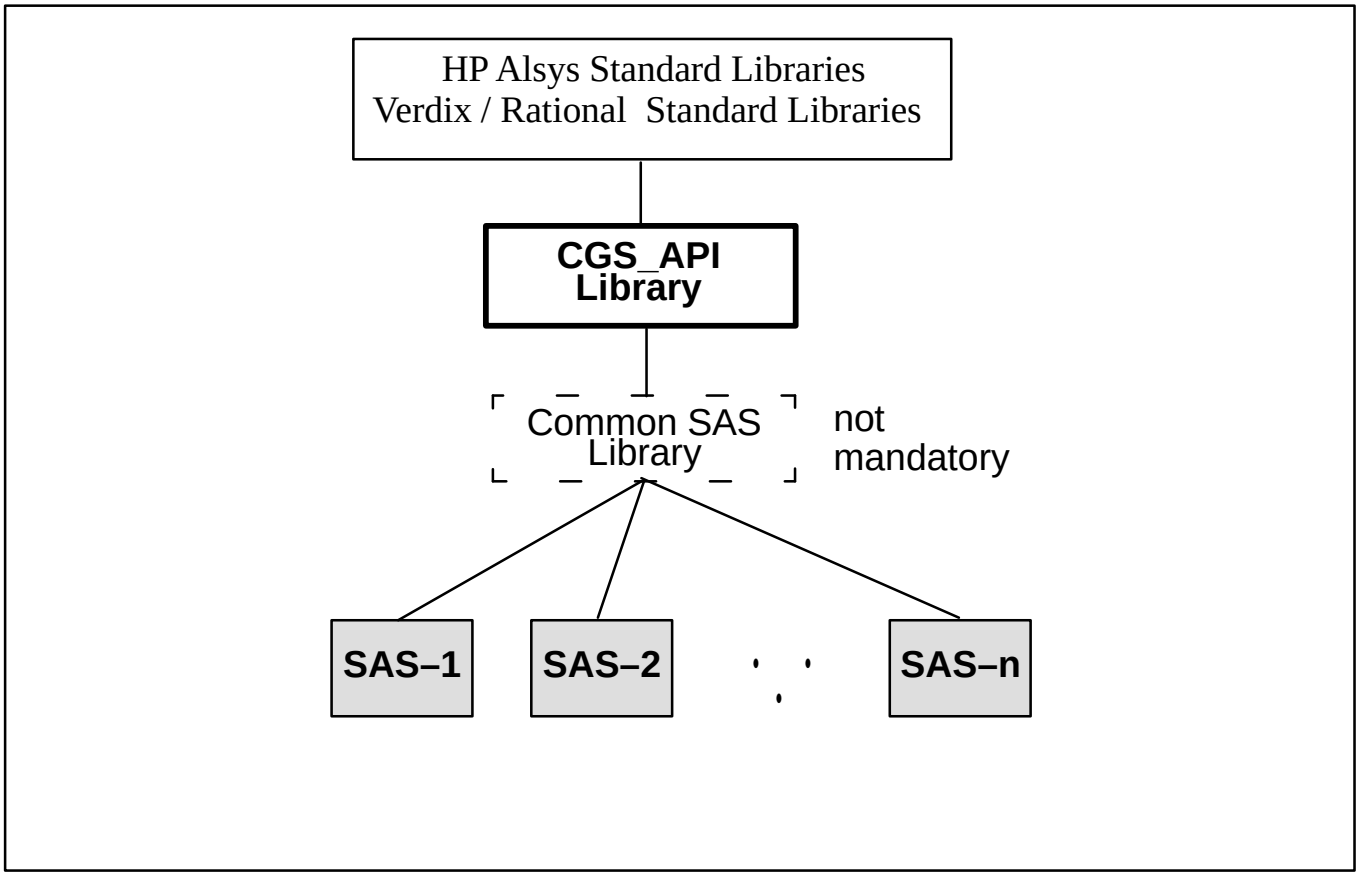


Figure 6–23 : *Ada Library Structure for SAS Development – this picture shows the visibility rule, not a real directory structure*

As already mentioned before, SAS constitute the real–time data and command interface to CGS via functions / procedures of packages in the TES_ API library or perform the specific calculations needed during on–line testing. Concerning the operational behaviour of the SAS, the main difference is whether or not it will internally contain ADA tasking or not. As such, the two different types of SAS are:

- Synchronous SAS, which wait for any CGS command in a permanent main loop inside the main program, then process it and after having processed it successfully, again start waiting for another command. These SAS could for example display raw data packets routed from other test nodes to workstations in a very special way or they could do special calibrations or limit checking of end item data.

  Synchronous SAS do not need Ada tasking inside the user written code (and we assume throughout this document that they will not contain Ada tasking!) since they sequentially perform a series of actions under CGS control.

- Asynchronous SAS, which have to handle requests from CGS as well as the front end equipment(s) in an asynchronous way do not know when and which event to handle. Usually, these SAS would include Ada tasking to handle the different events asynchronously. Possibly, UNIX interrupts are bound to task entries or special event handling routines (interrupt handler) are used.

Copyright per DIN 34

Asynchronous SAS are restricted to the limitations imposed by the Ada run-time environment provided with Alsys compiler. These restriction are mainly how asynchronous events have to be handled by the Ada run-time system and how Ada task scheduling is done.

In the following the main program structure of the two categories of SAS programs, synchronous and asynchronous will be explained in more detail. In order to understand the overall program logic it is important to note here that the TES API provides a procedure READ_COMMAND which returns a command from CGS to the SAS. Through this interface the SAS is controlled and all commands and data requests are routed to the SAS.

6.2.2 The SAS Main Program

The overall structure of a SAS 'main program' depends on the environment the SAS is intended for and also on the desired operational behavior.

In general, two different environments are envisaged:

- a non X–View environment for all SAS running on the HP node. Normally, these SASes constitute the real–time data and command interface to CGS via the product TES or perform the specific calculations needed during on–line testing. Additionally, those SASes running on a SUN platform without having an X–Views (or OPENLOOK) user interface also fall into this category.
- an X–View environment for those SASes running on the SUN, assuming these SASes have an own user interface using the X–VIEW and/or OPENLOOK style. In this case, the program has to include the so–called 'window main loop', a general purpose event handler provided by X/OPENLOOK. The main purpose of these SASes, as envisaged currently is the special visualisation of test data, special evaluations, etc.

Concerning the operational behavior of the SAS, the main difference is whether or not it will internally contain ADA tasking or not. As such, the two different types of SAS are:

- synchronous SASes, which in a permanent main loop inside the main program wait for any CGS command to come, then process it and after having processed it successfully, again start waiting for another command. These SASes could for example display raw data packets routed from other test nodes to workstations in a very special way or they could do special calibrations or limit checking of enditem data.
  These SASes do not need Ada tasking inside the user written code (and we assume throughout this document that they will not contain Ada tasking!) since they sequentially perform a series of actions under CGS control.
- asynchronous SASes, which have to handle requests from CGS as well as the front end equipment(s) in an asynchronous way not know when and which event to handle. Usually, these SASes would include Ada tasking to handle the different events asynchronously. Possibly, UNIX interrupts are bound to task entries or special event handling routines (interrupt handler) are used.

Not all possible combinations of the afore mentioned criteria will make sense due to the limitations imposed by the X–VIEW environment as well as the Ada runtime environment provided with a given compiler. These restriction are mainly how asynchronous events have to be handled by the Ada runtime system and how Ada task scheduling is done. Table 6–4 lists the allowed combinations and restrictions:

Copyright per DIN 34

	non X–VIEW	X–VIEW
synchronous	allowed, no restrictions	allowed, XVIEW restrictions
asynchronous	allowed, Ada run–time restrictions	allowed, XVIEW restrictions

Table 6–4 : *Restrictions in the ADA X–VIEW Environment*

6.2.2.1 General Structure in a Non–X–View Environment

In a non X–View environment, the Ada main program structure is only limited by the scheduling capabilities provided by the chosen compiler, i.e. the Ada run–time system, with respect to handling asynchronous IO.

In this chapter the main program structure of the two main categories of SAS programs, synchronous and asynchronous will be explained.

In order to understand the overall program logic it is important to note here that the TES API provides a procedure READ_COMMAND which returns a command from CGS to the SAS. Through this interface the SAS is controlled and all commands and data requests are routed to the SAS.

6.2.2.1.1 Synchronous SAS Main Program Structure

Synchronous SAS are those explicitly waiting for commands from CGS and then reacting to them without doing other processing in parallel (e.g. device IO). As such, the main program structure of this type of SAS could be as follows:

Synchronous SAS Main Program Structure

```
with TES_API;                                    — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                     — this package provides a high level interface for
...                                              —  handling the commands sent from TES to SAS
procedure MAIN is                                — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;  — the variable for the command

begin
      ...                                        — do some initializations
      loop                                       — the main program loop starts here
          ...                                    — do some more initializations
          TES_API.READ_COMMAND(...,              — wait for a command and block
                               COMMAND => CMD,
                               ...,
                               BLOCK => TRUE);
          process_command(...);                  — and process the command, e.g. the device IO
                                                 — occurs completely !
      end loop                                   — the main program loop ends here
end MAIN;                                        — here the main program ends
```

Copyright per DIN 34

This program really waits for a command from the CGS Test Execution Facility, e.g. a request to send a stimulus (GDU request) and then processes it, resulting in device IO. Only if all processing is finished, then the 'main loop' of the program starts from the beginning again, waiting for the next command.

The above kind of program logic does not allow for any periodical processing to take place inside the SAS, e.g. a cyclical IO with a device to be controlled. If this is desired, a pseudo asynchronous behaviour can be achieved using the following main program structure:

Pseudo asynchronous SAS Main Program Structure

```
with TES_API;                                   — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                    — this package provides a high level interface for
...                                             —  handling the commands sent from TES to SAS
procedure MAIN is
      — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;      — the variable for the command
DLY: duration;                                  — the overall cycle time for this SAS
```

to be continued...

```
begin
      ...                                       — do some initializations
      loop                                      — the main program loop starts here
          delay(DLY)                            — delay for the overall cycle time
          perform_cyclic_operations(...);       — do the regular job here
          TES_API.READ_COMMAND(...,                 — check for a command, do not block
                               COMMAND => CMD,
                               ...,
                               BLOCK => FALSE);
          if (ADT_TES_TO_SAS_COMMAND.COMMAND_ALTERNATIVE(CMD) <>
            ADT_TES_TO_SAS_COMMAND.NO_COMMAND) then
             process_command(...);              — process command if any
          end if;
      end loop                                  — the main program loop ends here
end MAIN;                                       — here the main program ends
```

Copyright per DIN 34

6.2.2.1.2 Asynchronous SAS Main Program Structure

If the handling of data inside the SAS has to be really asynchronous to the handling of the communication to TES, then another program logic has to be chosen, using the Ada tasking mechanisms.

A possible logic for this kind of application is the following:

Asynchronous TES SAS Main Program Structure

```
with TES_API;                                   — this package provides the TES_API operations !
with ADT_TES_TO_SAS_COMMAND;                    — this package provides a high level interface for
...                                             —  handling the commands sent from TES to SAS
procedure MAIN is                               — here the main program starts
CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;          — the variable for the command
DLY: duration;                                  — the overall cycle time for this SAS
task CMD_HANDLER is                             — specify a task to handle the commands
entry START;                                    — accept an entry to start the task
      ...                                       — possibly some more entries
end CMD_HANDLER;                                — end of this task specification

task PROCESSOR is                               — specify a task to process data asynchronously
entry START;                                    — accept an entry to start the task
entry DO_SOMETHING;                             — an entry to do some processing, e.g. a GDU
      ...                                       — possibly some more entries
end PROCESSOR;                                  — end of this task specification
```

to be continued...

Copyright per DIN 34

```
task body CMD_HANDLER is                          — body of the CMD_HANDLER task
begin
      accept START;                               — accept an entry to start the task
      ...                                         — do some initializations
      loop                                        — the main loop starts here
          TES_API.READ_COMMAND(...,               — check for a command, block
                               COMMAND => CMD,
                               ...,
                               BLOCK => TRUE);
          process_command(...);                   — process command, possibly rendezvous with
                                                  — PROCESSOR.DO_SOMETHING;
      end loop                                    — the main loop ends here
end CMD_HANDLER;                                  — end of the body for CMD_HANDLER

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

task body PROCESSOR is                                — body of the PROCESSOR task
begin
      accept START;
          — accept an entry to start the task
      ...                                         — do some initializations
      loop                                        — the main loop starts here
          select                                      — check which rendezvous is desired
              accept DO_SOMETHING do ...end accept;   — in case of DO_SOMETHING do it
              or
              accept ...                          — another rendezvous
          else
              do_something_else                   — no rendezvous, do something special,
                                                  — e.g. provide ADUs cyclically
          end select;
      end loop;                                   — the main loop ends here
end PROCESSOR;                                    — end of the body for PROCESSOR

– – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – – –

begin                                             — here the main program starts
      CMD_HANDLER.START;                          — activate the CMD_HANDLER
      PROCESSOR.START;
— activate the PROCESSOR
end MAIN;                                         — here the main program ends
```

6.2.2.2 General Structure in an X–View Environment

Deleted

Copyright per DIN 34

6.2.2.3 Handling Synchronous IO with Front End Devices

Synchronous IO with front end devices in this context means that the SAS takes the initiative and starts 'talking' to the device and then waits for it to respond. During this process, the SAS is blocked for other activities, i.e. communication with CGS.

This type of IO handling always works without problems in all environment, be it X–Views or UNIX solely. However, in the X–VIEW environment the general constraints of the window main loop have to be considered, as described already in section 6.2.2.2.

The general disadvantage of being blocked during the device IO, especially during the read from the external device, can be overcome by introducing asynchronous device IO, as it will be described in the next chapter.

6.2.2.4 Handling Asynchronous IO with Front End Devices

In asynchronous device IO mode, the SAS does not explicitly wait for device IO to occur but instead sets up a processing scheme internally in a way that IO can be handled whenever it occurs.

Usually, this is implemented using interrupt service routines, X–VIEW notifier procedures or Ada tasking. Which method can be used depends of course on the type of application the SAS is, i.e. whether it is an X–VIEW program or not, but also on the chosen compiler/runtime system. Many compilers do not correctly implement the asynchronous IO in the tasking environment.

Assuming the compiler vendor has delivered a runtime system being able of correctly handling asynchronous device IO, then the following general architectures of a SAS program exist:

- the SAS includes one or more tasks which are bound to device IO interrupts with some of their entries
- the SAS includes one or more tasks which are blocked in READs on the respective input lines
- the SAS includes X–VIEW notifier procedures which are invoked by the runtime system on the occurrence of an IO.

Copyright per DIN 34

6.2.3 How to Implement SAS

Before starting the implementation, a proper environment set-up has to be ensured in terms of setting-up

- the UNIX environment
- the compiler system environment
- the system topology table (for TES_SAS)

For the UNIX environment some variables have to be set and paths must be visible in the users search path. Some of the required conditions of the Ada compiler system environment has already been mentioned above, but the detailed data and additional information are given in the following procedure.

The system topology table is dependant on the environment at users site, therefore the set-up of the table is described in the installation manual.

Note that the SYSTEM_TOPOLOGY_TABLE is maintained by the **CGS administrator** only (See section on test setup using the TSCV tool).

6.2.3.1 Implementing a TES_SAS

deleted

6.2.3.2 Implementing a TEV_SAS

deleted

Copyright per DIN 34

6.2.4 How to Control TES SAS

TES SAS can be controlled either by APs containing the command sequence or interactive using the TES HLCL command line window.

A general description is given using the SAS state transition diagram.

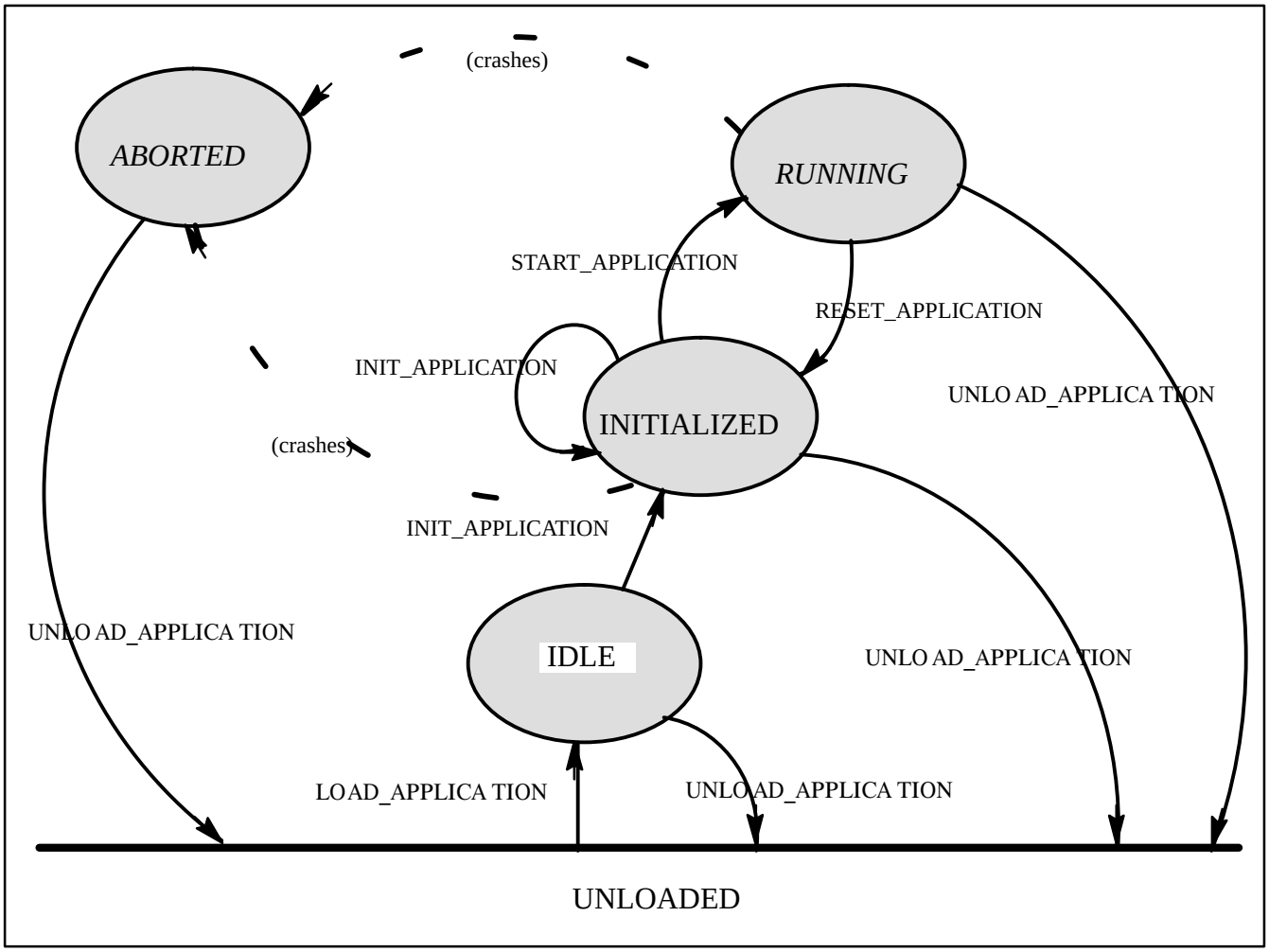


Figure 6–24 : *Relationship of SAS State Transitions*

TES SAS can be loaded (i.e. created as a UNIX process) via the UCL library command

- LOAD_APPLICATION

This command loads the SAS to the local test node or on a workstation. Also a Simulation Node or the DB Server node may be selected.

It is possible to transfer parameter to the SAS via the LOAD_APPLICATION command. These parameter are transfered on the command line and can be read by the SAS via the standard mechanisms (i.e. as if the parameter were given via a UNIX shell command).

Starting a SAS on a workstation may need to transfer within the parameter block the TES identification, where the SAS shall connect to: e.g.: ”<SAS Param1> TES_xx”

Copyright per DIN 34

The following constraints need to be followed:

- the TES identification must always be the identification of the test node where the SAS was started
- the SAS must not be started on another test node, where an entry exists in the System Topology Table for a TES_xx, independently, if a TES is running on this test node or not. (The TES_API will be confused, if the SAS is started on such a test node)

TES SAS can enter different states. After loading the SAS enters the state IDLE. The operator can now control the state of the SAS by giving the command

- INIT_APPLICATION

The SAS will initialize itself according to the parameter given and then enter the state INITIALIZED, if no serious error occurs (the SAS ”crashes”) , which prevents the SAS from initializing correctly. In this case the SAS enters the state ABORTED.
Normally the SAS will now remain in the state INITIALIZED until the

- START_APPLICATION

command has been received. The SAS will then enter the RUNNING state and perform its foreseen task. (If an error occurs the SAS goes to the state ABORTED.)

AS soon as the SAS is running commands requesting data can be send as well as commands to the UUT. (These commands with a description how to use them can be found in the UCL system library).
If the SAS is delivering ADUs, it will announce the ADU service after entering the RUNNING state. If the SAS is accepting GDUs, it will accept these in the RUNNING state only, after having announced the ADU service to TES. Also the GDU service will be announced in the RUNNING state only.

The SAS may be set back to the RESET state via the

- RESET_APPLICATION

command.

To remove the SAS from memory the

- UNLOAD_APPLICATION

command must be given.
In case of error the SAS enters the state ABORTED and must the removed from the test node using the UNLOAD_APPLICATION command as well.

Note that the sequence of state transitions is a guide-line only.
The SAS programmer is free to modify the SAS reactions upon the commands.
Example: The command INIT_APPLICATION *is not mandatory and may be omitted, if the programmer decides to do the initialisation after the* LOAD_APPLICATION *command. Nevertheless the programmer must guarantee that sending the* INIT_APPLICATION *command does not cause a SAS crash.*
Note that the SAS behaviour must be well documented and known as well to the programmer developing APs or operational procedures !

The current SAS state can be monitored by the operator on the SAS status display or using the

- GET_APPLICATION_STATUS

command which returns the status of an application including error status and statistics.

Copyright per DIN 34

Two UCL library procedures are available for the communication with SAS, they allow to read resp. to send a string from/to a SAS.
The UCL library command

- READ_MESSAGE_FROM_APPLICATION

additionally has a blocking indicator, which allows to specify if the procedure shall wait until an application really sends a message or if only the next message buffered is read.

- WRITE_MESSAGE_TO_APPLICATION

sends a string of up to 255 characters to the SAS.

Note that the contents of the messages to be exchanged depends on the implementation of the SAS!

Additionally the UCL library command

- DOWNLOAD

is provided. This commands downloads a file to a SAS, the corresponding frontend equipment managed by the SAS or the Unit Under Test (UUT) through a SAS.

Copyright per DIN 34

6.3 Preparing Ground Synoptic Displays

The graphic display editor is implemented by the ground window definition utility (GWDU). The GWDU is capable of editing so called synoptic displays. Synoptic displays are graphical representations of the test object. The display is animated by data delivered from the test object (e.g. a sensor represented by a thermometer displayed in the synoptic)

The user should refer to the GWDU User Manual (see reference 2.2.2.1) for details on use of this tool.

The GWDU tool is started via the I_MDB Tools menu on end items of type WDU_GROUND_SYMBOL and WDU_GROUND_SYNOPTIC_DISPLAY.

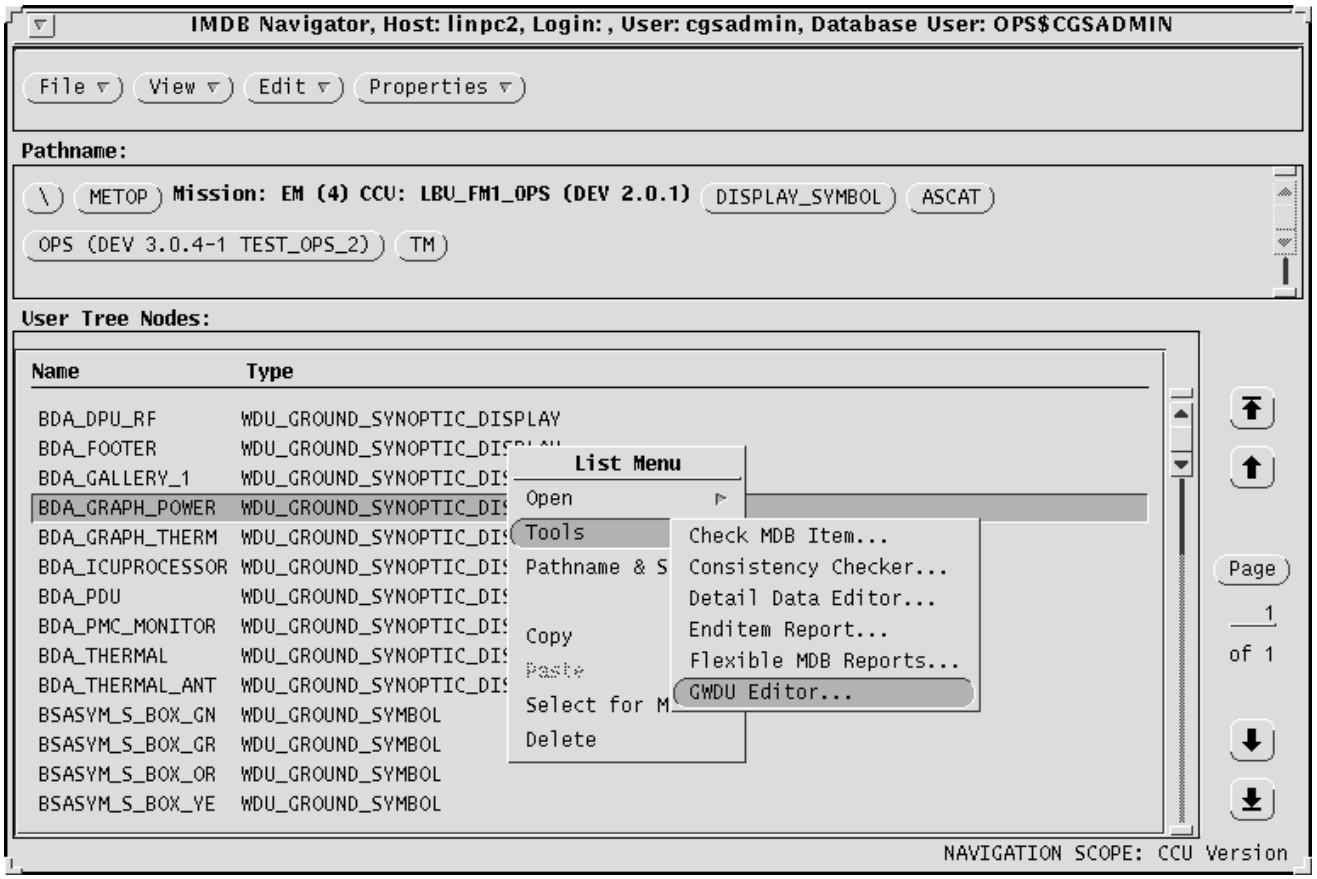


Figure 6–25 : *Starting the GWDU Editor from I_MDB Panel*

Copyright per DIN 34

6.4 Developing Simulation Models

This chapter describes the graphical model development language MDE–GL which is used in the Model Development Environment (MDE) to implement simulation models. The aim is not only to describe the different language elements and their use but also to gain an insight into the timing concepts of model execution. The latter is needed to understand the concepts of modelling and to develop CSS models successfully.

6.4.1 MDE–GL Language Elements

Figure 6–26 presents the MDE–GL lexical elements.

6.4.1.1 Composite Function Blocks

Composite function blocks (CFBs) are implemented graphically by MDE–GL. The various MDE–GL lexical elements, specifically interconnected atomic function blocks, parameter blocks and composite function blocks may be composed to one composite function block. The interface items of a CFB, which are accessible from outside and inside the function block, allow to draw connections into and out of a CFB.

Since the implementation of a composite function block may consist, among others, of further composite function blocks, composite function blocks constitute the nodes in the hierarchical (simulation) model function block tree structure. The root of this tree structure is the single top level composite function block (TLFB), up to 16 levels of decomposition are possible.

The top level composite function block represents the (simulation) model source, it is named with the model's name and its interface represents the simulation model's (external) interface. The interface items may reference onboard items (i.e. stimuli and measurements) in the MDB allowing to specify the connection of the simulation model (the simulator kernel) to an external system.

6.4.1.2 Atomic Function Blocks

Atomic function blocks are implemented by AIL code, either directly or by a decision table which is automatically transformed into AIL code. Each atomic function block can be seen as an AIL procedure, the specification of which is automatically derived from the MDE–GL graphical definition of its input/output interface: the outputs denote the in–out parameters while the inputs denote the in parameters. The incoming connections (ending at an input) and outgoing connections (starting at an output) denote the data flow.

Usually the AIL implementation code computes new output values based on input values passed to the function block, but also (old) output values may be accessed. An output of an atomic function block may be connected (finally) to inputs of atomic function blocks as well as to model (i.e. TLFB) outputs. An input of an atomic function block may be connected (finally) to an output of an atomic function block, the output of a parameter block or a model (i.e. TLFB) input.

There are two different types of atomic function blocks: Synchronous atomic function blocks (SFBs) are activated periodically with the user specified time frame. Asynchronous atomic function blocks (AFBs) are activated by events, i.e. whenever an activation event occurs for at least one of its inputs (see section 6.4.2).

6.4.1.3 Parameter Blocks

A parameter block (PB) keeps one output. In contrast to atomic function blocks, parameter blocks are never executed and therefore the value of the output stays constant, unless it is changed by the user via an assignment.

6.4.1.4 Interface Items

Three kinds of interface items are provided: Inputs, outputs and grouping links.

Copyright per DIN 34

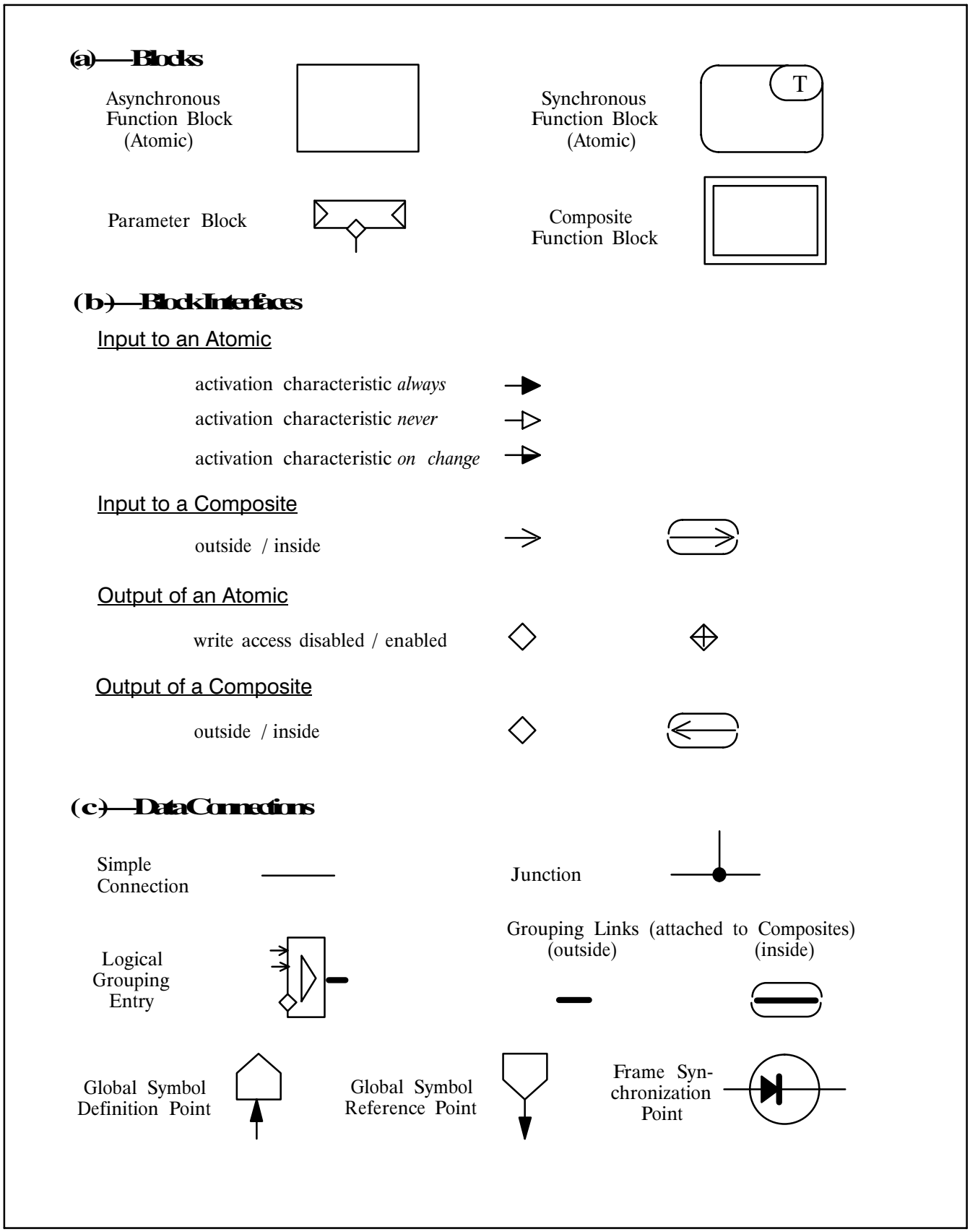


Figure 6–26 : *MDE Graphical Language Lexical Elements*

Copyright per DIN 34

The inputs and outputs of an atomic function block denote the in and in–out parameters of the corresponding AIL procedure. The output of a parameter block represents a constant value. Inputs and outputs of a composite function block allow to draw connections into and out of the composite function block. The inputs and outputs of the top level composite function block (TLFB) denote the external interface of the simulation model. Inputs and outputs of grouping entries represent the signals contained in the logical grouping.

The outputs of atomic function blocks and parameter blocks and the model (TLFB) inputs represent the data sources, whereas the inputs of atomic function blocks and model (TLFB) outputs represent the data sinks. Each data sink must be connected to a data source.

Each input and output is associated with a particular data type chosen from a predefined set of data types. Connections between interface items are restricted to items of compatible type.

Inputs of atomic function blocks are marked with an activation characteristic. Outputs of atomic function blocks may be marked for write access (i.e. to enable the user to set the value of the output during simulation execution).

Grouping links may be attached to composite function blocks, allowing to draw logical groupings into and out of them.

6.4.1.5 Simple Connections

Simple connections allow to connect the various outputs and inputs (of atomic function blocks, composite function blocks, parameter blocks and grouping entries) in the MDE–GL implementation of a particular composite function block together or to global symbols, thereby specifying the data flow.

Also grouping entries may be connected together or to grouping links at the interface of composite function blocks.

6.4.1.6 Logical Groupings

A logical grouping is a bus of signals, each signal has a unique name within the grouping and can be used to connect interface items. A logical grouping is built by a set of interconnected grouping entries. Signals are accessed by the inputs and outputs of the grouping entries; each signal consists of one input at one of the grouping entries and of one or more outputs at other grouping entries connected. The input and the outputs representing a particular signal are named with the same name, i.e. the name of the signal; they can be connected to the outputs and inputs of function blocks, the outputs of parameter blocks or to global symbols. Grouping links allow to draw a logical grouping into and out of composite function blocks.

Grouping entries connected to the TLFB interface represent a special case: The number of inputs and outputs that can be attached directly to the interface of the TLFB is restricted by the extent of the graphic. In order to avoid any limitation on the number of inputs and outputs at the simulation model external (i.e. TLFB) interface, grouping entries may be connected to a grouping link at the interface of the TLFB. The outputs of these grouping entries represent model inputs, the inputs of these grouping entries represent model outputs. Here, the signal mechanism (i.e. one signal input, several signal outputs) is not supported, grouping entries connected to the TLFB interface may not be interconnected with ordinary logical groupings as described above.

6.4.1.7 Global Symbols

A global symbol provides another means for connecting interface items. It consists of one definition point and one or more reference points. Each reference point is connected to the definition point with the same name, found in the same composite function block as the reference point or in the next higher level composite

Copyright per DIN 34

function block with respect to the line of CFBs, beginning with the CFB which contains the reference point straight up to the TLFB. The definition point and the reference points can be connected to the outputs and inputs of function blocks resp. grouping entries and to the outputs of parameter blocks.

6.4.1.8 Frame Synchronization Points

The purpose of the Frame Synchronization Point is todelay a signal's propagation to a receiver (usually an asynchronuous FB) until the start of the following simulation time frame.

Thus, FSPs are the tool to split asynchronuous chains and, especially, loops (see section 6.4.2.3).

Note: In the current implementation, inputs of AFBs with activation ***never*** are treated as if connected via an implicit FSP, i.e. they will receive changed values from their senders in the following simulation time frame.

Editorial Note:
Since the Frame Synchronization Point has been added only recently, most screen snapshots of the model editor window in this chapter have not been updated to include the corresponding icon.

6.4.2 Model Execution Strategy

The execution of the atomic function blocks is based on a time frame concept. The basic time frame is called the simulator minframe. All other time frames are multiples of the minframe.

6.4.2.1 Input Activation Modes

AFBs and hibernating SFBs have to be marked for activation in order to become executed. This is done when a specific activation event occurs for at least one of the FB's inputs. Three different input activation characteristics are provided:

- *always*
  The FB is marked for activation if the parent FB of the output connected to the input has been executed; if the input is connected to a model input, it is marked for activation if the referenced stimulus has been written by the external system.
  In addition, the FB is marked for activation, if an assignment issued by the user to the output resp. model input connected to the input or to the input itself has occured.
- *on change*
  The FB is marked for activation if the value of the output resp. model input connected to the input has changed, i.e. has been updated with a value different from its value before the update. Such an update may result from a calculation of an output during execution of the output's parent FB, from a write operation of the external system to a stimulus referenced by a model input or from an assignment issued by the user.
  In addition, the FB is marked for activation, if an assignment issued by the user to the input itself, that changed the input's value, has occured.
- *never*
  Such an input has no impact on activation of its parent FB.

Note that the occurence of such an event does not lead to an immediate execution of the FB, instead the FB is marked for activation and will be executed synchronized with its specific time frame. Also, the occurence

Copyright per DIN 34

of multiple events before the FB is actually executed does not lead to multiple execution, a number of events greater than 1 is handled the same way as just 1 event.

Inserting a FSP into a signal will delay not only thecorresponding value update, but also the activation for a receiving AFB until the following simulation time frame. If the AFB also has non–synchronized inputs, this means that it may be activated twice in two consecutive frames.

A special semantic is associated with the pulse types (i.e. PULSE resp. BURST_PULSE). An update of an interface item of a pulse type (resulting from a calculation of an output during execution of the output's parent FB, from a write operation of the external system to a stimulus referenced by a model input or from an assignment issued by the user) is referred to as a trigger. This indicates that, in contrast to all other data types, the value does not persist until the next update, for each AFB or SFB input it is reset automatically when the execution of the parent FB has been completed. Therefore each update (trigger), even consecutively with the same value, marks all connected FBs (in the case of a trigger of a model input resp. of an AFB, SFB or PB output) resp. the parent FB (in the case of a trigger of an AFB or SFB input) for activation. AFB and SFB inputs of pulse type must be marked with the activation characteristic *on change*.

6.4.2.2 Synchronous Function Blocks

Unless hibernating, SFBs are executed periodically with a time frame specified as a multiple of the simulator minframe. Each execution period lasts the specific SFB's time frame. The function code may include an AIL statement to suspend the cyclic activation temporarily, causing the SFB to hibernate. A hibernating SFB has to be marked for activation before it is executed periodically again.

An SFB is executed exactly once per execution period. For each SFB, the exact start and end times of its execution periods with respect to the start of the simulation, i.e. the first minframe (minframe 1) are fixed for the entire simulation: The n–th execution period starts with the beginning of the ((n–1) * T + 1)–th minframe and ends with the (n * T)–th minframe where T is the SFB's time frame (duration of the execution period) in multiples of the minframe. The first execution period of each SFB starts with the beginning of the simulation, i.e. with the beginning of the first minframe. During execution, an SFB may set itself hibernating, this means that it will be suspended temporarily for execution (in its following execution periods) as long as it is not marked for activation (see 6.4.3.1). Suspension and reactivation of an SFB are synchronized with its time frame (i.e. they have no impact on the execution period scheme described).

The value of an output of an SFB becomes available (to connected inputs of SFBs, to connected inputs of AFBs and to connected model (TLFB) outputs) at the end of the SFB's execution period.

If the activation criteria are met, hibernating SFBs connected to an output of an SFB are marked for activation at the end of the SFB's execution period, they will be executed in their usual next execution period. AFBs connected to an output of an SFB are marked for activation also at the end of the SFB's execution period, they will be executed in the following minframe.

For each execution, the inputs of an SFB are parameterized with the values which are available at the beginning of the SFB's execution period; outputs of an SFB retain their last computed value. Since SFBs operate on input value snapshots made at the beginning of the execution period, they can be seen as parallel processes that are independent from activation orders.

Copyright per DIN 34

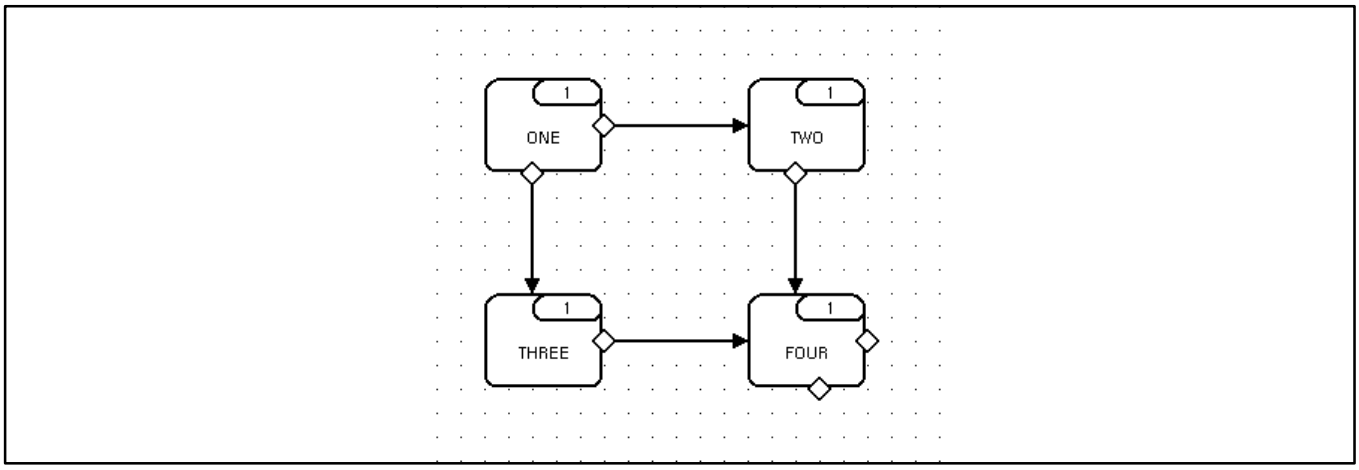


Figure 6–27 : *A number of SFBs connected together*

Figure 6–27 shows some connected synchronous function blocks. All function blocks are executed *concurrently* within their time frame.

Synchronous function blocks may be used to trigger the execution of asynchronous groups (see next section).

6.4.2.3 Asynchronous Function Blocks

AFBs connected together in any direction (i.e. not restricted to forward direction with respect to the arrow symbols of interface items) via at least one input of activation characteristic *always* and/or *on change* belong to the same asynchronous group. The AFBs in a group are serialized for execution, the order is defined by the connections between the interface items. A correct serialization is any total ordering of the group member FBs which complies with the following rule:

> Any group member FB_i is executed after all group members FB_j with a connection from an output of FB_j to an input with activation characteristic *always* or *on change* of FB_i (a necessary and sufficient precondition is the acyclicity of the asynchronous group).

Depending on the situation more than one order may be legal. CSS transforms the graphical model specification into a linear program structure, automatically choosing one of several possibilities, if necessary.

Each asynchronous group is processed as a whole during one minframe. Processing means that, one by one in a correct order, each AFB that is marked for activation is executed (each AFB marked for activation is executed exactly once, the execution of an AFB may mark a succeeding AFB of the same group for activation); AFBs not marked for activation are skipped.

The value of an output of an AFB becomes available a) to connected inputs of SFBs, to connected inputs of AFBs of activation characteristic *never* and to connected model (TLFB) outputs at the end of the minframe the AFB is currently executed in and b) to connected inputs of AFBs of activation characteristic *always* resp. *on change* immediately, i.e. in the current minframe, unless the connection is interrupted by a FSP, in which case the value becomes available in the following minframe.

If the activation criteria are met, hibernating SFBs connected to an output of an AFB are marked for activation at the end of the minframe the AFB is currently executed in, they will be executed in their usual next execution period. AFBs connected to an output of an AFB are marked for activation immediately, they will be executed in the current minframe.

For each execution, the inputs of an AFB are parameterized with the values which are available at the time of the beginning of the AFB's execution; outputs of an AFB retain their last computed value.

Copyright per DIN 34

Figure 6–28 shows an asynchronous group triggered by an SFB running every minframe. During the first minframe none of the AFBs is executed (no computations are performed) because none of the AFBs has been marked for activation at this time. At the end of the first minframe, the AFB ONE is marked for activation by SYNC. During the second minframe the complete asynchronous group will be processed (when the asynchronous group of this example is processed, always all AFBs are executed because all inputs are of activation characteristic *always*).

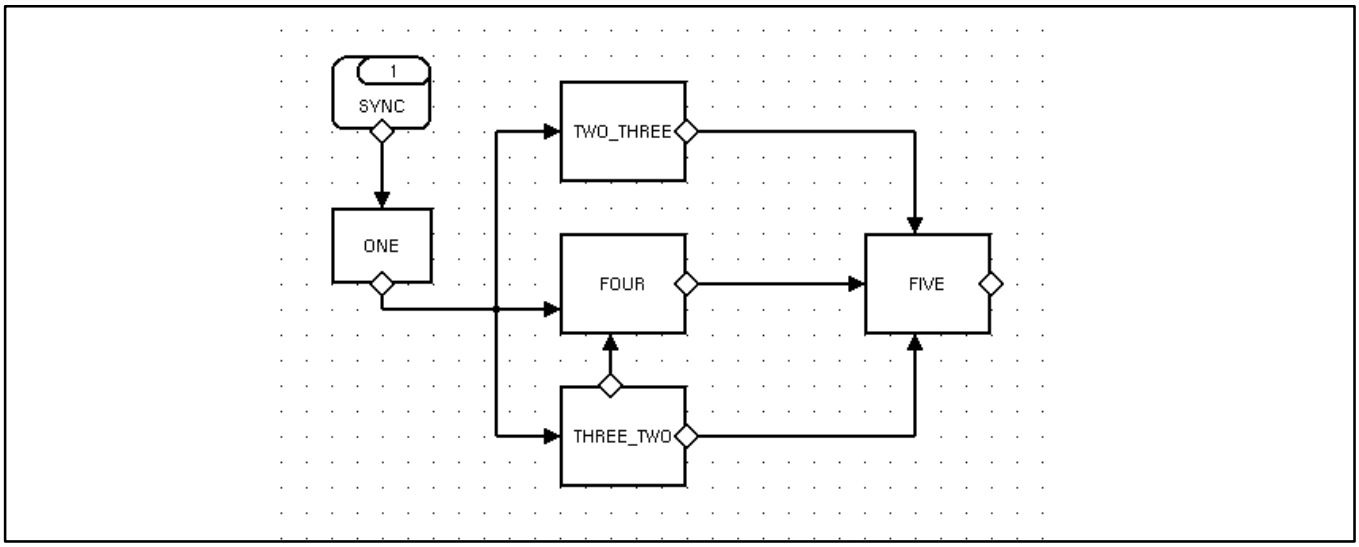


Figure 6–28 : *An example of an asynchronous group triggered by a synchronous function block*

The graphical specification allows 2 serializations, both are legal:

SYNC: –> ONE –> TWO_THREE –> THREE_TWO –> FOUR –> FIVE or
SYNC: –> ONE –> THREE_TWO –> TWO_THREE –> FOUR –> FIVE

Note that the AFB FOUR will be executed once (with the relevant values from ONE and THREE_TWO) although it is marked for activation twice (arrows from FB ONE and FB THREE_TWO).

The user can not determine whether the function block TWO_THREE or the function block THREE_TWO will be executed first.

The situation becomes more complicated if the user changes some of the input activation characteristics to *on change* (the parent FB will be activated if the predecessor FB connected to the input has changed the value of the output connected to the input). Both serializations are still valid but it is possible that the execution of single function blocks is skipped.

Figure 6–29 shows a case where two SFBs of different time frame are used as triggers in order to reduce computation load during model execution.

Copyright per DIN 34

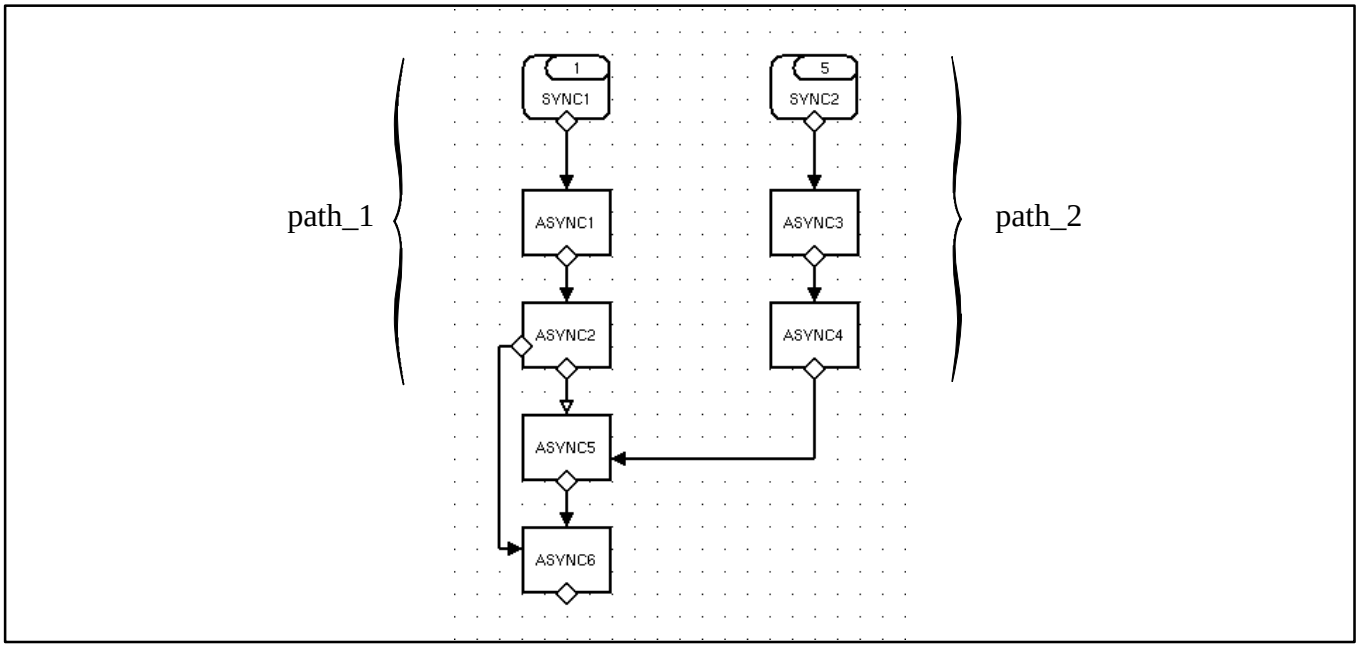


Figure 6–29 : *Use of two synchronous function blocks to reduce system load*

The AFBs all belong to the same asynchronous group, but the AFBs in path_1 and also ASYNC6 will be executed every minframe (the SFB SYNC1 is running every minframe) whereas the AFBs in path_2 and also ASYNC5 will be executed every fifth minframe. Note that if the connection between ASYNC2 and ASYNC6 would be missing or the input of ASYNC6 connected to the output of ASYNC2 would be of activation characteristic *never*, there would be two different asynchronous groups (ASYNC1–ASYNC2 and ASYNC3–ASYNC4–ASYNC5–ASYNC6).

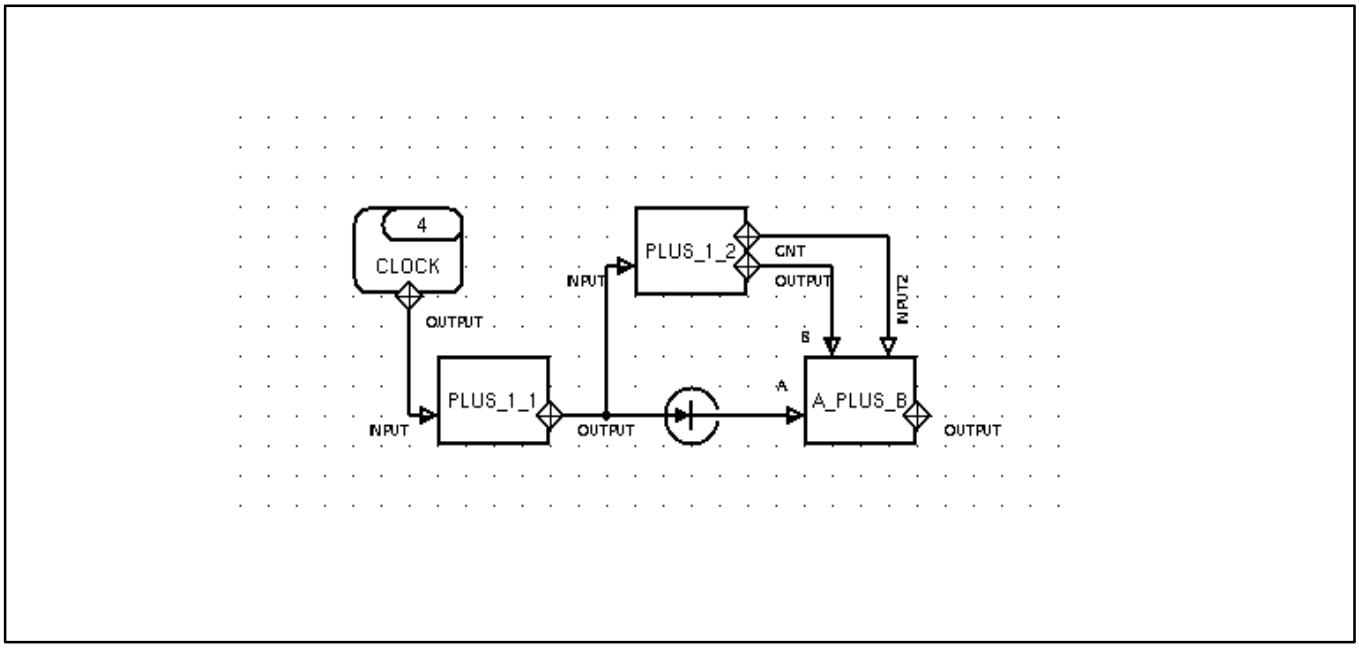


Figure 6–30 : *Asynchronuous chain with FSP and activation* ***never***

Copyright per DIN 34

In Figure 6–30, we finally see an asynchronuous chain with FSP and one input with activation mode ***never.*** Note that this is just an example for explanation of the activation mechanisms, not for good modelling style!

The output of SFB CLOCK triggers AFB PLUS_1_1 every 4 minframes. This will immediately activate AFB PLUS_1_2 (via INPUT), followed by A_PLUS_B (via B, assuming their input values have changed). Due to the FSP connected to input A and mode ***never*** (with implicit FSP) of input INPUT_2, A_PLUS_B will receive the just calculated value of OUTPUT from PLUS_1_2 as input B, but the previous values of OUTPUT from PLUS_1_1 as A and of CNT from PLUS_1_2 as INPUT_2.

In the following minfarme, due to the FSP, the delayed update of OUTPUT from PLUS_1_1 will activate A_PLUS_B once again, now with the updated values for A and INPUT_2 from the previous minframe.

6.4.2.4 Connection to an external system (H/W in the loop)

If a simulation model (a simulator kernel) is connected to an external system (which sends stimuli to and receives measurements from the model) its timing behaviour is still determined by the frame machine concept. Model inputs are treated in the same way as outputs, model outputs are treated in the same way as inputs of synchronous function blocks with an execution period of 1 minframe.

The minimum model response time resulting from this timing behaviour is 2 minframes (of course, if the implementation of a simulation model requires multiple minframes to perform the computations of the model outputs based on values passed for the model inputs by the external system, the model response time increases accordingly).

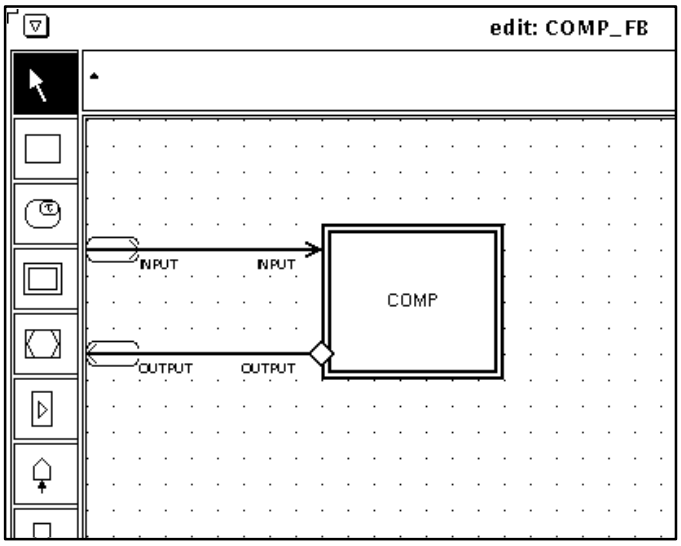


Figure 6–31 : *A model with external interface (one model input and one model output).*

Another consequence of the described timing mechanism is that only the last of a series of multiple events occurring within one execution period will take effect – all previous events will be lost. This may happen if events generated by the external system are processed by model functions executed with a slower frequency than the external event frequency.

6.4.2.5 The Simulation State

The simulation state contains the complete dynamic state information of a simulation model at a given time during a simulation execution, it comprises the current values of all atomic function block outputs, parameter

Copyright per DIN 34

block outputs and model inputs, the current values of all atomic function block inputs and model outputs as well as the current function block activation states.

When a simulator kernel is configured, a simulation state that covers the initial model state information is automatically generated. It is based on the initial values of the atomic function block outputs, parameter block outputs and model inputs and the initial running/hibernating states of synchronous function blocks as entered by the model developer using the MDE tools.

Copyright per DIN 34

6.4.3 Implementation Of Atomic Functions

As mentioned before, atomic functions must be implemented either by decision tables or using AIL code (AIL – Atomic Implementation Language, defined as an Ada subset). AIL covers the elementary Ada features necessary for the implementation of atomic function blocks, as arithmetic and logic operations, conditional branching and loops.

Certain Ada features are explicitly forbidden for the implementation of atomic functions to avoid conflicts with the simulator kernel. In practice, this means that certain Ada keywords (see 6.4.3.1) are rejected by the Atomic Implementation Editor, and that the usual Ada standard library packages are inaccesible. The forbidden Ada features (e.g. tasking or standard I/O packages) are not needed for the implementation of atomic functions: All necessary system programming functions for parallelization, synchronization, function activation etc. are completely specified by the MDE–GL model definition.

Local variables may be declared in which case their lifetime is limited by one execution of the associated function (i.e. there are no static variables). Furthermore, local subprograms may be declared and defined. No object declared inside the code of one function block can be made visible to another function block (i.e. there are no global variables), and no function block implementation can call the code of any other function block directly. This ensures that the MDE–GL graphical connections are the only means of communication between model functions.

Each atomic function block can be seen as an Ada procedure, the specification of which is automatically derived from the MDE–GL graphical definition of its input/output interface (cf. Figure 6–32).

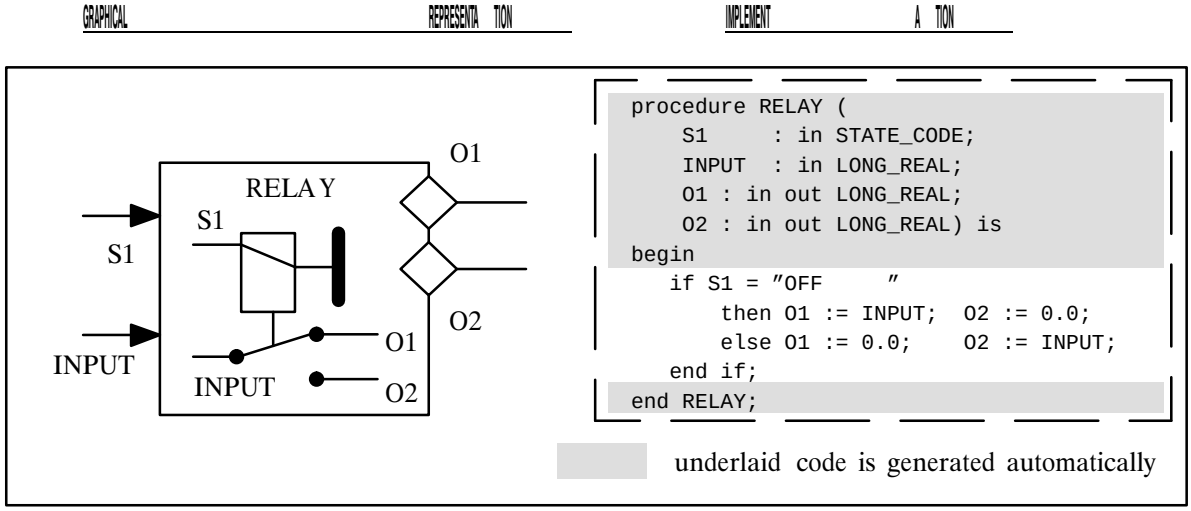


Figure 6–32 : *MDE-GL Atomic Function Block with AIL Implementation*

The parameters to such a procedure are the following:

- one (Ada–) in parameter for each function block input, initialized with the value which is available at the beginning of the AFB execution resp. SFB execution period
- one (Ada–) in out parameter for each function block output, initialized with the current value of this output variable (i.e. the value computed during the last execution of this function, or the initial value when the function is activated for the first time)

6.4.3.1 Description of AIL

The Ada subset covered by AIL encompasses:

Copyright per DIN 34

– the declaration of types (excluding task– and access types), constants and variables
– standard arithmetic and logical operations
– standard control structures, such as branches and loops
– the definition of subprograms (functions and procedures)
– the inheritance of certain standard library functions, e.g. math functions

Figure 6–32 illustrates the use of AIL for the implementation of atomic function blocks and its automatic integration with graphical MDE–GL function block definitions.

Identifiers

The syntax of AIL identifiers is equal to Ada identifier syntax, i.e. identifiers may consist of letters (a..z, A..Z), digits (1..9) and the underline character. An identifier must start with a letter and may not end with an underline; the consecution of two underlines is ruled out.

Predefined Visible Declarations Common To All Atomic FB Implementations

The following standard and CSS specific Ada declarations are implicitly visible to the atomic FB implementation code (AIL or decision table conditions/actions). At the same time, they are the *only* visible declarations:

Standard (Predefined) Types; Literals

The following standard data types as defined by the Ada LRM (3.5.2–3.5.4, 3.5.7, 3.6.3) are provided:

```
BOOLEAN   INTEGER   FLOAT   CHARACTER   STRING
```

Literals of these types are written as described in the Ada LRM 2.4–2.6.

Examples:

```
TRUE           FALSE                        -- boolean literals

0              +20           -14362         -- integer literals
0.0            +20.0         -0.123         -- real literals
1.23E-12       +1.0E+6       -4.67E-2       -- real literal with exponent

2#1111_1111#   16#FF#        016#0ff#       -- literal for 255, expressed
                                            -- for different bases

'a'            'X'           '@'            -- character literals
"hello"        "Hi there!"   "!@#$%^&*"     -- string literals
```

CSS Specific (Predefined) Types

CSS provides the following additional types to be used as types for function block inputs/outputs and for the declaration of local variables. However, not all of these types may be assigned to the inputs/outputs of a simulation model (i.e. top level composite function block); in the following listing the types that may be assigned also to model interface items are underlined. The types are contained in the Ada package CSS_TYPES, the declarations are implicitely visible to atomic function block implementation code (note that the type BOOLEAN, although a standard predefined type, is listed here again for the reason of completeness, since it may be assigned to function block inputs/outputs):

– UNSIGNED_BYTE
  is an integer type with the range 0..+2**8–1.

Copyright per DIN 34

– SIGNED_BYTE
is an integer type with the range –2**7..+2**7–1.

– UNSIGNED_SHORT_WORD
is an integer type with the range 0..+2**16–1.

– SIGNED_SHORT_WORD
is an integer type with the range –2**15..+2**15–1.

– UNSIGNED_INTEGER
is an integer type with the range 0..+2**31–1 (note that 2**31–1 is not a typographical error).

– SIGNED_INTEGER
is an integer type with the range –2**31..+2**31–1.

– REAL
is a floating point type in 32 bit IEEE format (IEEE single float). To inputs and outputs an engineering unit can be assigned; type compatibility checking between inputs and outputs is extended to identical engineering unit.

– LONG_REAL
is a floating point type in 64 bit IEEE format (IEEE double float). To inputs and outputs an engineering unit can be assigned; type compatibility checking between inputs and outputs is extended to identical engineering unit.

– BOOLEAN
is actually a standard predefined type.

– STATE_CODE
is an ”enumeration type” with user selectable representation; type compatibility checking between inputs and outputs is extended to identical representation. Actually it is a string type of 8 characters, each string must comply with AIL identifier syntax, possibly filled up with trailing blanks (<space> signs).

Example:

```
OUTPUT := ”ON      ”;   -- statecodes: ”OFF     ”, ”ON      ”
```

– TIME
is a private type the internal representation of which is hidden; there are no literals. The following functions and procedures may be used to assign resp. access variables:

```
subtype YEAR_NUMBER is INTEGER range 1901..2099;
subtype MONTH_NUMBER is INTEGER range 1..12;
subtype DAY_NUMBER is INTEGER range 1..31;
subtype DAY_DURATION is LONG_DURATION range 0..86400.0;

function TIME_OF (YEAR: YEAR_NUMBER;
     MONTH: MONTH_NUMBER;
     DAY: DAY_NUMBER;
     SECONDS: DAY_DURATION := 0.0) return TIME;
function YEAR (DATE: TIME) return YEAR_NUMBER;
function MONTH (DATE: TIME) return MONTH_NUMBER;
function DAY (DATE: TIME) return DAY_NUMBER;
function SECONDS (DATE: TIME) return DAY_DURATION;
procedure SPLIT (DATE: in TIME;
     YEAR: out YEAR_NUMBER;
     MONTH: out MONTH_NUMBER;
     DAY: out DAY_NUMBER;
     SECONDS: out DAY_DURATION);
```

Copyright per DIN 34

TIME includes a function CLOCK which returns the current simulated mission time (SMT). Additionally a function GET_LOCAL_TIME is provided which returns the current local time.

Examples:

```
OUTPUT := CLOCK;                  -- current SMT

OUTPUT := GET_LOCAL_TIME;         -- current local time

OUTPUT := TIME_OF (1997, 12, 24);-- december the 24th, 1997
```

– LONG_DURATION
is a fixed point type with the range –86.400.0..+86.400.0 seconds. It includes a function GET_MINFRAME_INTERVAL which returns the increment in simulated mission time (SMT) per minframe. The syntax for literals is equal to the syntax of REAL literals.

– COMPLEX
is a record type. The two components RE and IM, both of type LONG_REAL, denote the real and imaginary part of a complex number in Gauss notation.

Examples:

```
OUTPUT := (10.0, –5.0);      -- set real part to 10.0, imag. part to –5.0

OUTPUT.RE := 10.0;           -- set real part to 10.0
OUTPUT.IM := –5.0;           -- set imaginary part to –5.0
```

– PULSE
allows to trigger a pulse output (on the sender side) and to test if a given pulse input was triggered. It provides a simple trigger mechanism for cases in which only event propagation, not a concrete value is relevant. The predefined constant PULSE_TRIGGERED is provided for triggering (assignment) and testing a variable in AIL code.

Examples:

```
OUTPUT := PULSE_TRIGGERED;                              -- trigger
if (INPUT = PULSE_TRIGGERED) then ... end if;           -- test
```

– BURST_PULSE
allows to trigger a burst pulse output (on the sender side) and to test if a given burst pulse input was triggered. It provides a trigger mechanism that comprises the transfer of information, i.e. the number of pulses. The predefined constant BURST_PULSE_NOT_TRIGGERED is provided for testing a variable in AIL code. The syntax for literals is equal to the syntax of UNSIGNED_INTEGER literals.

Example:

```
OUTPUT := 3;                                            -- trigger
if (INPUT /= BURST_PULSE_NOT_TRIGGERED) then            -- test
     if INPUT > 5 then ... end if;
end if;
```

– Various VECTOR types:
UNSIGNED_BYTE_VECTOR, SIGNED_BYTE_VECTOR,
UNSIGNED_SHORT_WORD_VECTOR, SIGNED_SHORT_WORD_VECTOR,

Copyright per DIN 34

UNSIGNED_INTEGER_VECTOR, SIGNED_INTEGER_VECTOR,
REAL_VECTOR, LONG_REAL_VECTOR,
BOOLEAN_VECTOR,
COMPLEX_VECTOR
Inputs/outputs of VECTOR types must consist of at least 2 elements, the maximum VECTOR size is restricted to 255 elements.

Examples:

```
OUTPUT := (8.1, –17.7, 4.5);-- REAL_VECTOR of 3 elements
OUTPUT(1) := 8.1;
OUTPUT(2) := –17.7;

X: BOOLEAN := INPUT(5);     -- access 5th element of a BOOLEAN_VECTOR
```

– Various MATRIX types:
UNSIGNED_BYTE_MATRIX, SIGNED_BYTE_MATRIX,
UNSIGNED_SHORT_WORD_MATRIX, SIGNED_SHORT_WORD_MATRIX,
UNSIGNED_INTEGER_MATRIX, SIGNED_INTEGER_MATRIX,
REAL_MATRIX, LONG_REAL_MATRIX,
BOOLEAN_MATRIX,
COMPLEX_MATRIX
Inputs/outputs of MATRIX types must consist of at least 2 columns and 2 rows (i.e. 4 elements), the maximum MATRIX size is restricted to 255 (i.e. columns * rows) elements.

Examples:

```
OUTPUT := ((7.3, –2.5, 4.0), (1.5, 3.9, –2.7));-- REAL_MATRIX of 2 rows
OUTPUT(1,1) := 7.3;                            -- and 3 columns
OUTPUT(2,3) := –2.7;

X: BOOLEAN := INPUT(5,3);            -- access element in 5th row and
                                     -- 3rd column of a BOOLEAN_MATRIX
```

– RECORD
The RECORD type provided for function block outputs and inputs allows to simplify the MDE–GL graphics: Instead of creating multiple outputs at one atomic FB and the corresponding number of inputs at another atomic FB and connecting each of them, it is possible to create one output and one input of RECORD type and to connect them by a single connection. This RECORD type is not available in the AIL implementation code; each component of such an output/input is treated as an individual scalar output/input, accessed by its name which consists of the concatenation of the output resp. input name, the underline character and the component name. Such a RECORD may consist of the following scalar types:
UNSIGNED_BYTE, SIGNED_BYTE,
UNSIGNED_SHORT_WORD, SIGNED_SHORT_WORD,
UNSIGNED_INTEGER, SIGNED_INTEGER,
REAL, LONG_REAL,
BOOLEAN,
COMPLEX

Copyright per DIN 34

Example:

```
OUTPUT_X := 2.3                  -- an output named OUTPUT of RECORD
OUTPUT_Y := 7;                   -- type consisting of three components:
OUTPUT_Z := TRUE;                -- X (REAL), Y (UNSIGNED_INTEGER) and
                                 -- Z (BOOLEAN)
```

User Defined (Local) Types

The user may define additional types, e.g. subtypes of predefined types (subtypes are useful if the user's abstraction of the problem space defines subsets of a base type, as e.g. a range constraint of allowed temperature values) or new enumeration types, to improve the readability of the atomic implementation code. Also array and record types may be declared. The visibility of such type declarations is limited to the scope of a single atomic function block within which they are declared (i.e., user defined types are strictly local to a function block). The syntax of these types is defined by the Ada LRM (3.3.2, 3.5.1, 3.6, 3.7).

Examples:

```
type temperature  is range 1.0 .. 2_000.0; -- a type declaration
subtype SMALL_INT is INTEGER range –10 .. 10;
type RELAY_POSITION  is (OFF, ON);         -- an enumeration type declaration
type TABLE is array (1..10) of INTEGER;    -- a simple array type declaration
type BUFFER is                             -- a simple record type declaration
     record
          POS: INTEGER := 0;
          VALUE: STRING (1..100);
     end record;
```

Since user defined types are always declared locally inside the implementation code of an atomic function block, only predefined types may be used for a block's inputs or outputs.

Local Constant and Variable Declarations

Local constants and variables of any standard, predefined or user defined type may be declared, invisible outside the function block in which they are declared, and with a lifetime limited by one execution of the corresponding function.

Examples:

```
PI       : constant := 3.1459_26536;      -- a real number
ONE, EINS: constant := 1;                 -- two different names for 1
MESSAGE  : constant STRING
               := ”ERROR IN FUNCTION F1”;  -- a constant string
COUNT, SUM    : INTEGER;                  -- a variable declaration
```

Expressions

Expressions may be formed as defined by the Ada LRM (chapter 4, excluding 4.8). The following examples are legal AIL expressions (provided that all shown variables are declared to be of appropriate types):

Copyright per DIN 34

Examples:

```
4.0                          -- real literal
SUM                          -- variable
INTEGER'LAST                 -- attribute
SINE(X)                      -- function call
COLOR'(BLUE)                 -- qualified expression
(LINE_COUNT + 10)            -- parenthesized expression
-4.0 + A                     -- compound expression
B**2 - 4.0*A*C               -- compound expression
X = 3                        -- relation
COUNT in SMALL_INT           -- relation
```

Statements

An AIL statement may be one of the following:

– the `null` statement

– a simple statement:
an assignment, `return`– or `exit` statement, or a procedure call

– a block:
non–empty list of statements (each followed by ';') enclosed in
`begin ... end`, resp. `declare ... begin ... end`

– a conditional statement:
`if ... then ... end if`
`if ... then ... else ... end if`
`if ... then ... elsif ... end if`
`case ... is when ... => ... when others => ... end case`

– a loop statement:
`loop ... end loop`
`while ... loop ... end loop`
`for ... in ... loop ... end loop`
`for ... in reverse ... loop ... end loop`

Syntax and semantics of these statements are as defined by the Ada LRM (chapter 5).

Examples:

```
OUT1 := COS(PI);                      -- assignment statement

if IN1 < 0 then OUT1 := 5;            -- if statement
else OUT1 := 6;
end if;
```

User Defined (Local) Subprograms

The user may define local functions and procedures. These can be called only from within the code of the function block they are defined in. The syntax for functions and procedures is described in the Ada LRM (chapter 6).

Copyright per DIN 34

Examples:

```
function MAX (X, Y: LONG_REAL) return LONG_REAL is
begin
     if X > Y then
          return X:
     else
          return Y;
     end if;
end MAX;
```

```
procedure SWAP (X: in out REAL; Y: in out REAL) is
declare
     T: REAL;
begin
     T := X;
     X := Y;
     Y := T;
end SWAP;
```

Math Package

The Ada package CSS_MATH provides a variety of reliable and reusable mathematical subprograms. In addition it defines data types, and numerical and physical constants, see section 6.4.11. The declarations are implicitely visible to atomic function block implementation code.

Hibernating Synchronous Function Blocks

Each synchronous function block implicitely has a variable named HIBERNATE of type BOOLEAN. The function block can set itself into a hibernating state (i.e. it will be suspended temporarily for execution starting with the following execution period until it is reactivated by an activation event) by assigning the value TRUE to this variable in the AIL code. Note that the current execution period is always completed, i.e. statements eventually following the hibernating statement are still performed.

Example:

```
if ... then
     HIBERNATE := TRUE;              -- start hibernating
end if;
```

Message Output

Two procedures are provided allowing to output messages from AIL. The first procedure is specified as follows:

```
type MESSAGE_TYPE is
     (INFO_MESSAGE, WARNING_MESSAGE, NON_CRITICAL_ERROR, CRITICAL_ERROR);
procedure ERROR_MESSAGE (MESSAGE: STRING;
     FUNCTION_BLOCK_NAME: STRING;
     TYPE_OF_MESSAGE: MESSAGE_TYPE := NON_CRITICAL_ERROR);
```

It allows to write a message (parameter MESSAGE) to the MOCS console window. The parameter FUNCTION_BLOCK_NAME should be set to the name of the atomic function block producing that message.

The second procedure makes use of the message file containing all the messages used by the simulator kernel and is specified as follows:

Copyright per DIN 34

```
procedure ERROR_MESSAGE (MESSAGE_NUMBER: INTEGER;
     REPLACE_STRING: STRING;
     FUNCTION_BLOCK_NAME: STRING);
```

It allows to write a message to the MOCS console window, to the log file and to the Test Result Data Base (TRDB). If CSS has been started from HCI, the message can also be sent to the CGS Error Services (i.e. the CGS Message Handler window), depending on a flag associated with the message in the message file. The parameter MESSAGE_NUMBER identifies the message in the file. The parameter REPLACE_STRING contains the values to replace the place holders in the message, if there are any. If there is more than one place holder to be replaced, the values have to be separated by the <tab> sign (i.e. ASCII.HT) or the '@' character in the replace string. The parameter FUNCTION_BLOCK_NAME should be set to the name of the atomic function block producing that message.

The name of the message file is specified by the environment variable CSS_KERNEL_MESSAGES. The contents of this file is split into 5 sections labeled by keywords. One of these sections comprises messages to be sent from AIL, it is labeled by the keyword $CSS_AIL_MESSAGES. Here, the user can define messages in the following syntax:

No<tab>*CGSI*<tab>*Crit.*<tab>*Message*

No is the integer index allowing the parameter MESSAGE_NUMBER to address the respective message. The *CGSI* flag may be either Y or N indicating whether the message shall also be sent to the CGS Error Services (Y means yes, N means no). However, this directive is performed only if CSS has been started from HCI.
Crit. may be either I, W, N or C indicating the criticality of the message (INFO_MESSAGE, WARNING_MESSAGE, NON_CRITICAL_ERROR, CRITICAL_ERROR). Within the CGS Error Handler window, I and W are transfomed to ADVISORY, N corresponds to ORDINARY and C corresponds to SEVERE. *Message* is the message string, it may contain place holders *%V1%*, *%V2%*, etc. that are replaced by the values in the parameter REPLACE_STRING.

The following example results in the message `This is an example!!!` that is written to the MOCS console window, to the log file, the TRDB, and, if possible, (i.e. if CSS has been started from HCI) is sent also to the CGS Error Services.

Example:

```
# entry in message file:
6913 Y    I    This %V2% an %V1%!!!

-- corresponding AIL call in Asynchronous FB \DEMO\RELAIS_2:
ERROR_MESSAGE (6913, "example" & ASCII.HT & "is", "\DEMO\RELAIS_2");
```

Forbidden Ada Features

All of the following Ada keywords are forbidden (thereby eliminating the features associated with these keywords):

```
abort, accept, access, at, delay, entry, generic, pragma, select, separate,
task, terminate, with
```

This means, that e.g. address, length, and representation clauses are ruled out, as well as the unportable and unsafe use of pragmas. The MDE Atomic Function Editor will reject any use of the above keywords inside atomic implementation code.

Since only those declarations which are made implicitly visible (standard and predefined, see above) are known to the implementation code, the normal Ada standard library units such as SYSTEM, TEXT_IO, UN-

Copyright per DIN 34

CHECKED_CONVERSION or UNCHECKED_DEALLOCATION are completely inaccessible (deliberately, to avoid conflicts with the simulator kernel).

Atomic Function Implementation Rules

The following rules summarize the constraints for AIL programming, which are checked in part by the MDE rule checking function and in part by the Ada compiler system. They apply to functions implemented completely using AIL code as well as to the definition of decision table conditions and actions:

- Atomic FB implementation code must not contain Ada keywords which are related to tasking, dynamic memory, exception handling, and type representation.
- The use of input variables (formal parameters) and output variables inside the implementation code of atomic function blocks must match their specified type.
- System calls are not allowed for the implementation of atomics.

6.4.3.2 Atomic Implementation by Decision Tables

For decision tables a table editor is provided that ensures the completeness and consistency of the table by construction.

A decision table consists of two parts: a set of conditions and a set of actions. A condition is an expression of type BOOLEAN (i.e. two–valued).

Conditions and actions are linked by a decision matrix which specifies a column for each possible combination of conditions. To reduce the table size, *'no matter'* values for conditions may be used. Further, a column labelled *'others'* is automatically generated to catch all combinations of conditions which are not explicitly specified. A second part of the matrix associates each column with a (possibly empty) set of action(s) to be executed whenever the corresponding combination of conditions is met.

An action is either (a) an executable AIL statement or (b) an assignment. In case (a), the statement is written in the action field, and all table columns for which this action shall be executed must be marked. In case (b), the action field of the table contains the name of a function block output, while the value to be assigned must be specified in the respective table column. If for an action the table column for a specific combination of conditions is left blank, the corresponding action is not executed (a) resp. no assignement is made (b).

All conditions are evaluated before any actions are executed. With the exception of this restriction, the evaluation order of conditions and the execution order of actions is arbitrary.

Code entered for conditions and actions must be correct AIL resp. Ada code with the restrictions of section 6.4.3.1.

A sketch of a decision table shown by the Decision Table Editor is given in Figure 6–33. The user may edit the shown decision table, which is initially empty. The Decision Table Editor will ensure by construction that the decision table is complete and consistent at all times.

The user may define local variables and macros to abbreviate expressions and statements. Local variables are always computed at the beginning of the function block execution, they may be read accessed in conditions and statements. A macro allows simple textual substitution. The syntax is:

name=substitution_text

The symbol *name* is replaced in conditions and actions by the arbitrary text *substitution_text*. The substitution text is the rest of the line following the equal sign, macro definitions over multiple lines are not possible. The macro expansions are performed in the order the macros are specified, i.e. it is possible to use macros in substitution texts, see the example.

Copyright per DIN 34

name field		condition combination columns							
									others
condition fields			condition matrix fields						
action fields			action matrix fields						

Figure 6–33 : *Sample outline of a decision table*

Examples:

```
MUL3=MUL2 * INPUT_3
MUL2=INPUT_1 * INPUT_2
```

The number of conditions is restricted to 8; otherwise a decision table may become too large to be handled easily.

Atomic Function Implementation Rules

The following rules summarize the constraints for AIL programming, which are checked in part by the MDE rule checking function and in part by the Ada compiler system. They apply to the definition of decision table conditions and actions:

- Decision table conditions must represent valid AIL resp. Ada expressions.
- Decision table actions must represent syntactically correct AIL/Ada statements.

Copyright per DIN 34

6.4.4 Model Development Pre–requisites

Within I_MDB the creation and modification of nodes within the element configuration tree shall not be performed by any user, but only by personal with MDB CM access. A model developer does not have MDB CM access in general.
Therefore a CSS user should assure that following items are already created in the MDB before starting a model editing session.

Following MDB items are needed

* a CCU which references at least one CDU used for model development
* a CDU of domain CSS for the models

*Note that the MDB user who performs the model development must be the **owner** of the CDUs.*

This has to be checked first before a model editing session can be started successfully.

To connect the model to the 'outer world' at least one additional CDU is required. It must contain the stimuli (commands) to be sent to and the measurements to be received from the model.

For the first model development steps and model testing purposes this CDU is not mandatory.

6.4.4.1 Starting a Model Editing session

The fist action to do is to invoke the appropriate CGS tool supporting the mission configuration phase. This tool is known as I_MDB (Interactive Mission Database access) and will provide the user with a window called 'I_MDB Navigator'.

Starting a Mission Configuration Session

- **Move** the **mouse cursor** to the menu ***Select Task*** of the ***CGS Task Selector*** window.
- **Hold the right mouse bottom**. You get now a menu with all tasks you may select. (see Figure 6–34).
- **Move** the **mouse cursor** to the task ***Mission Preparation*** and **release** the **right mouse button**.

Note, the contents of the task list is not fixed and can be modified for each user by editing the '.task_list' file. Figure 6–34 shows the standard CGS Task Selector

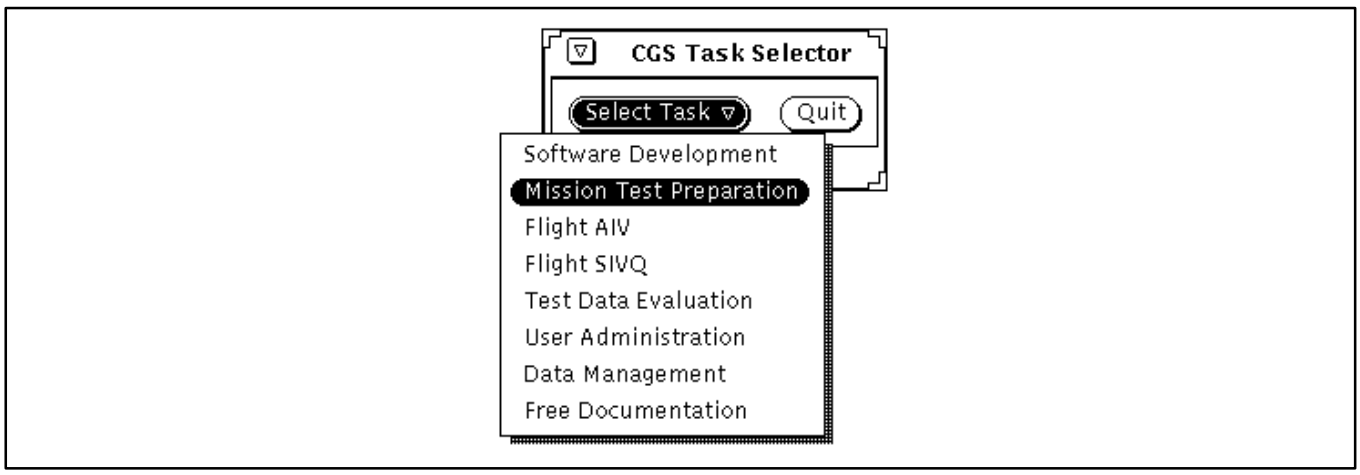


Figure 6–34 : *Mission Configuration Start from CGS Task Selector*

Copyright per DIN 34

Select the System Tree version

Navigating down within a System Tree

- *Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.*
- **Double click** on the element you want to navigate to. This results to a box called ***System Tree Versions*** listing all available version of the element system tree and the corresponding CM status.
- **Double click** on the version you want to navigate to. Now the I_MDB Navigator window is updated as follows :
  - The Current-Path shows the Element Name and its system tree version.
  - The Node-List provides a list of all system tree nodes forming part of this version

Select the CCU

By selecting a CCU version a special view on the element configuration tree is created, providing only those CDUs contained in the selected CCU version. This makes it easier to find a specific CDU in the MDB.

A CCU can be selected either from a complete list of all CCU versions defined for the selected element or from a list of all CCU versions defined for a particular System Tree node :

Navigating down to a CCU version from the complete list

- **Move the mouse** to the Menu-Line and **select File–>Browse All CCU Versions...** . This opens the ***Browse CCU Versions*** box listing all defined CCUs including pathname, CCU Name and CCU version.
- **Use the Scroll-Bar** on the right site of ***Browse CCU Versions*** if necessary.
- **Click** on the CCU you want to access and **click** on the ***Apply*** button.

For more information about navigating to a CCU refer to chapter 5.1.2.3.

Select the CDU

For the procedure below it is assumed that a system tree version of an Element Configuration has be accessed and a CCU has been selected which contains the desired CDU.

Navigating down within a User Tree Version

- *Note, that the use of the Scroll-Bars may be necessary, but is not explicitly mentioned.*
- **Double click** on the items in the tree structure until you reach the desired CDU.
- **Double click** on the CDU to be selected. This action leads to the I_MDB Navigator window which now lists :
  - The path to the CDU version as the Current-Path
  - A list of models forming part of the CDU version in the Node-List

Copyright per DIN 34

Create a new model (Toplevel Function block)

In the beginning the selected CDU is empty i.e. there is no list of models. The users first action is to create a top–level function block (this is the name for the highest level of a model in the MDB terminology).

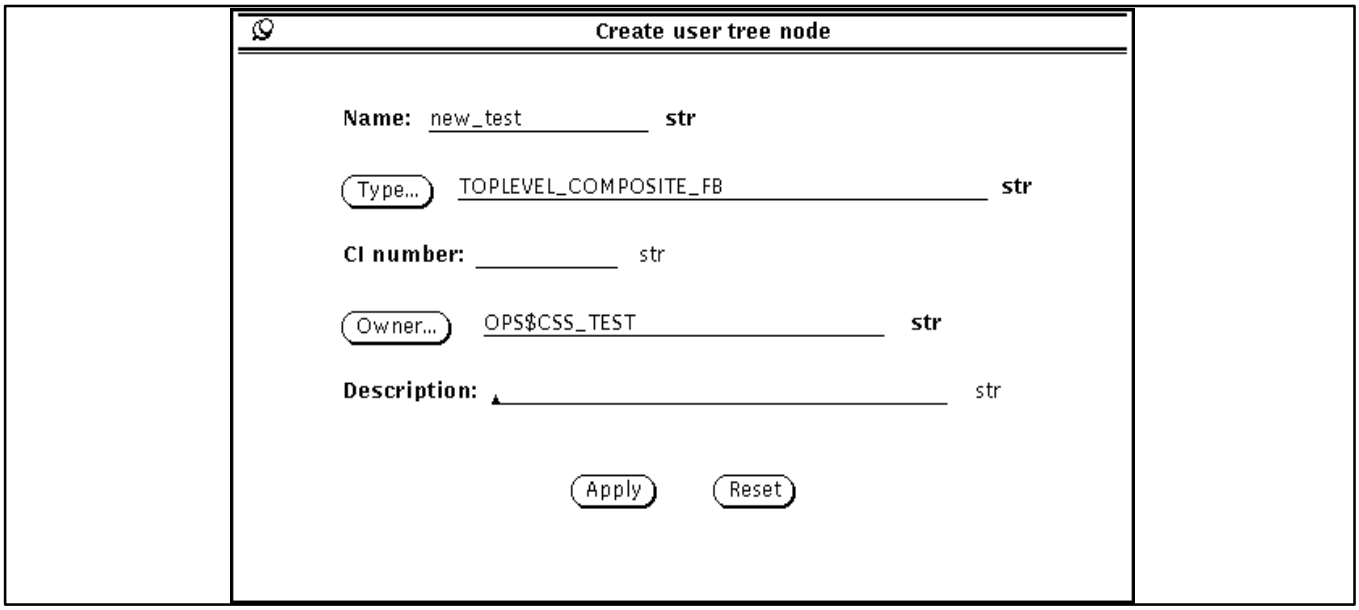


Figure 6–35 : *The Create user tree node window*

Creating a new model

- Press the **File** button with the left mouse button and select **Create Node...** from the pull–down menu. The ***Create user tree node*** window appears. (see Figure 6–35)
- Type the **model name** in the name field.
- Press the **Type** button. The ***Node Type list help*** window pops up. (see Figure 6–36)
- Select the type **TOPLEVEL_COMPOSITE_FB** and press the **Apply** button.
- Type a **model description** in the description field.
- Press the **Apply** button. The new model name appears in the I_MDB window.

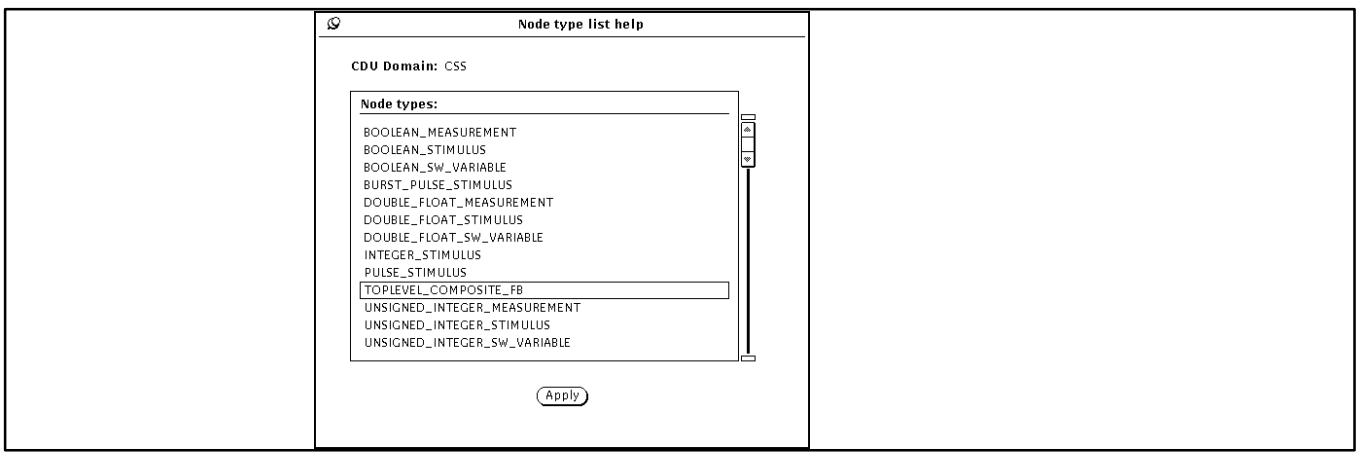


Figure 6–36 : *The Node type list help window*

Copyright per DIN 34

6.4.5 Starting CSS User Interfaces

When invoked from I_MDB, CSS will automatically open the pre–selected simulation model for editing or execution. Additionally CSS presents itself with the data base browser (DBB) for browsing through CSS relevant data in the data base (f.e. CDUs containing CSS models), and finally routing to MDE or MOCS after having selected data for editing resp. execution and observation.

Summarizing CSS provides the following three user interfaces:

* DBB (Data Base Browser)
* MDE (Model Development Environment)
* MOCS (Model Observation and Control System)

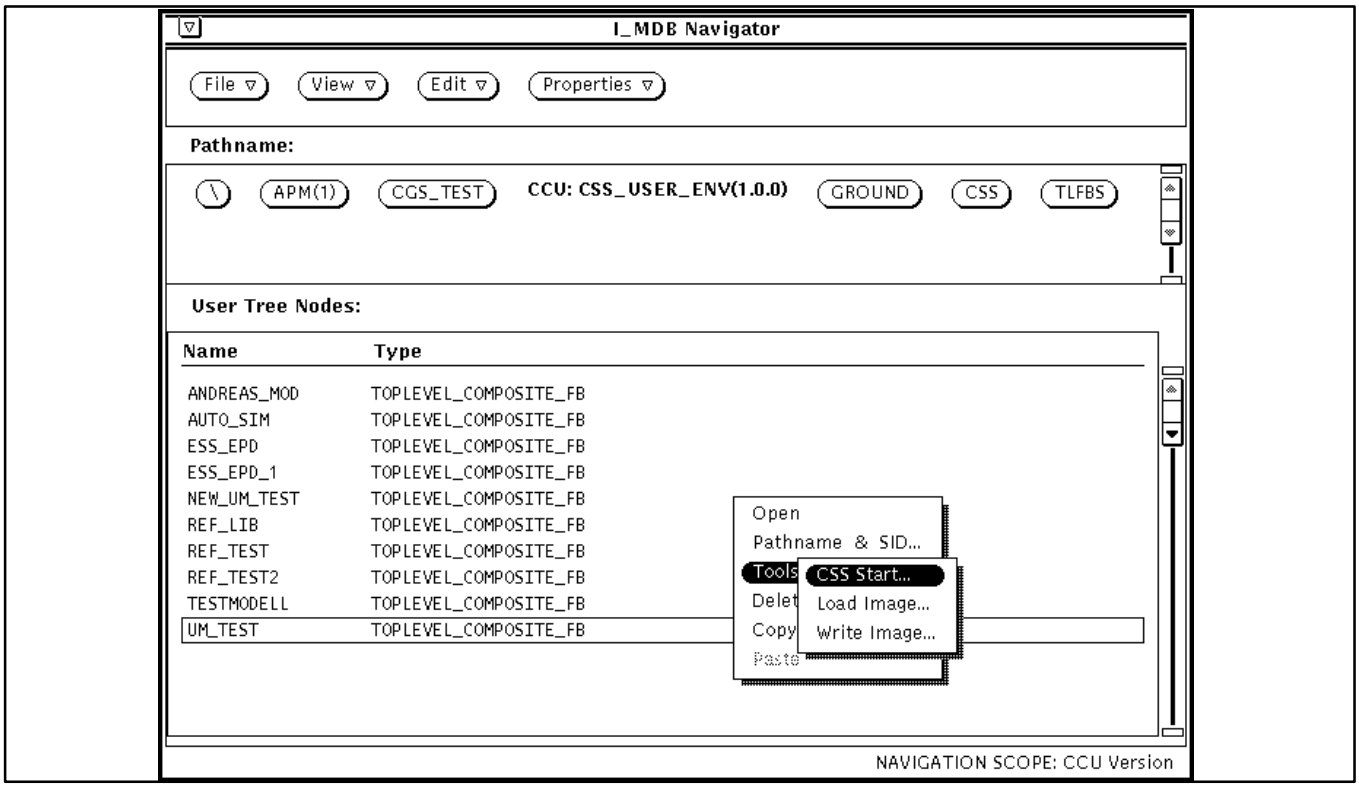


Figure 6–37 : *The CDU with a list of models*

Start the CSS User Interface for Model editing

Start the CSS User Interface for Model editing

- **Select** the model in the I_MDB window. (see Figure 6–37)
- Press the left mouse button and select **CSS start...** from the pop–up menu.
- The CSS scope check window appears. MDE is pre–selected. Scroll to the bottom of the message area to see the latest appended message. If no unresolved references are reported you can press the **Start CSS...** button.
- The Data Base Browser window and the Composite Editor window will be opened automatically.

Copyright per DIN 34

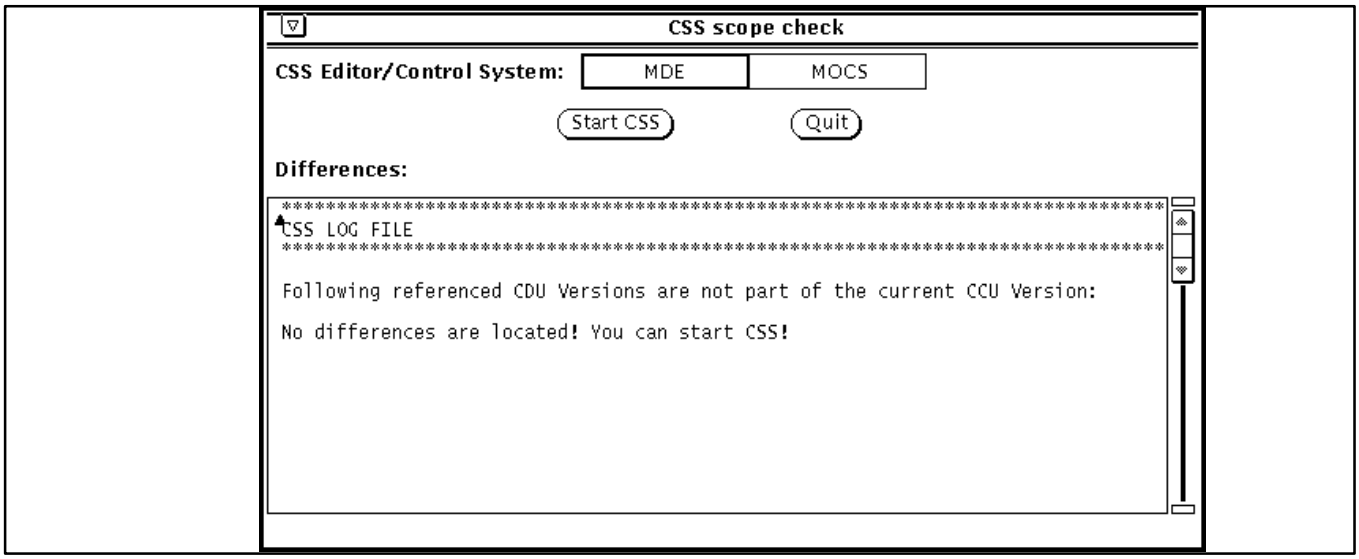


Figure 6–38 : *The CSS scope check window*

Start the CSS User Interface for Model editing with unresolved references

Start the CSS User Interface for Model editing

- **Select** the model in the I_MDB window. (see Figure 6–37)
- Press the left mouse button and select **CSS start...** from the pop–up menu.
- The CSS scope check window appears. MDE is pre–selected. Scroll to the bottom of the message area to see the latest appended message. The CSS scope check window reports the missing CDU versions (see Figure 6–39). Nevertheless you can press the **Start CSS...** button.
- A decision window pops up (see Figure 6–40). If you decide to open the model, press the **yes** button.

Note that updating the invalid function references may be a really time consuming procedure

- The Data Base Browser window and the Composite Editor window will be opened automatically.

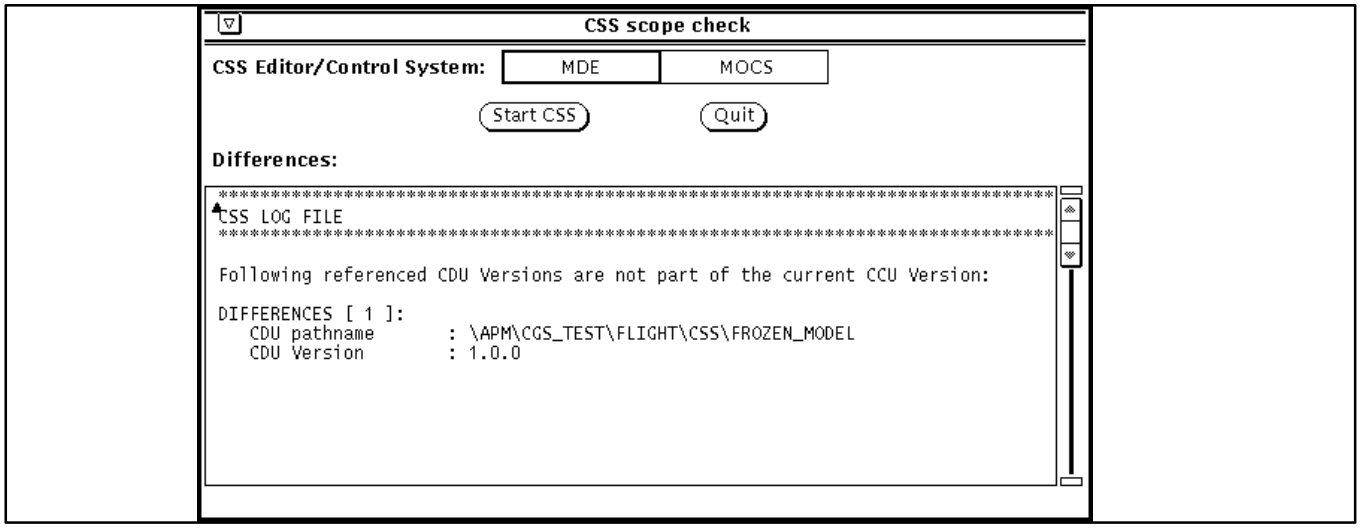


Figure 6–39 : *The CSS scope check window reports missing references*

Copyright per DIN 34

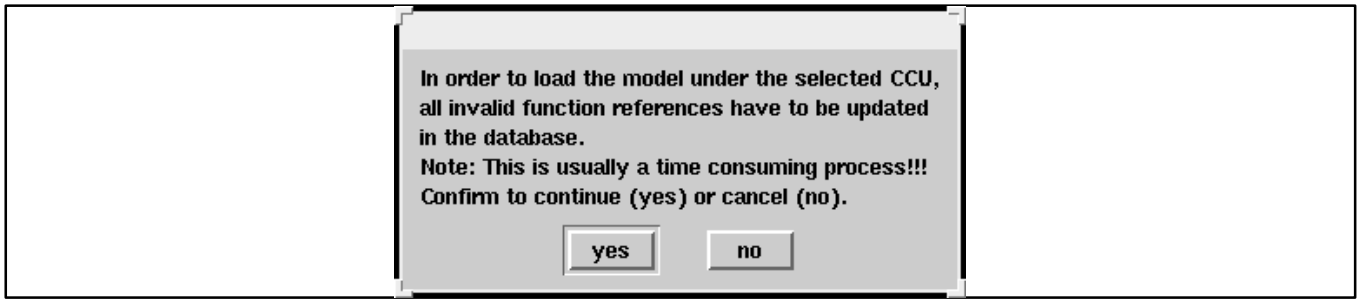

In order to load the model under the selected CCU,
all invalid function references have to be updated
in the database.
Note: This is usually a time consuming process!!!
Confirm to continue (yes) or cancel (no).
yes
no

Figure 6–40 : *The CSS decision window*

6.4.5.1 Restrictions on model editing

The following functions are not yet available in the current implementation.

- The necessary functions to use CSS type ”COMPLEX” are not yet implemented.

Copyright per DIN 34

6.4.6 Database Browser User Interface

6.4.6.1 General

The DBB is a subsystem of the CSS User Interfaces which allows to browse through the underlying database both for simulation models and for definitions of onboard items (i.e. stimuli and measurements).

For simulation models, CSS tools for inspecting and/or editing simulation models, configuring executable simulator kernels (MDE), inspecting and/or editing simulation tables, starting, controlling and monitoring simulations (MOCS) may be invoked.

Definitions of onboard items may be selected and exported to the relevant MDE tools (i.e. to create the references from simulation models to onboard items).

6.4.6.2 DBB Master Window

The DBB user interface consists of a single master window, tiled into several subviews.

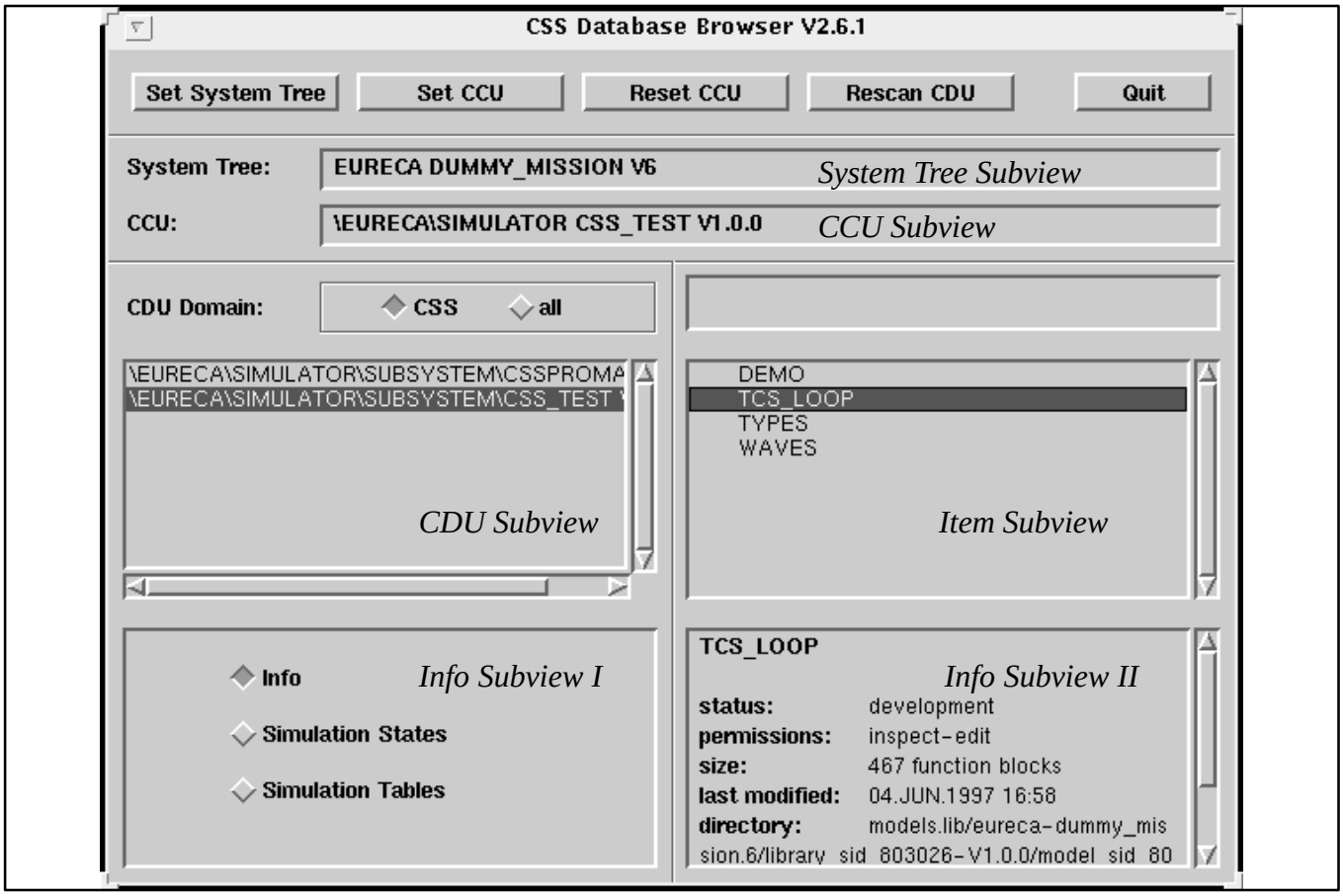


Figure 6–41 : *DBB Master Window with selected model*

The DBB forms the central part of the CSS User Interfaces (i.e., it controls the execution of all other tools); it is the only tool which is continuously present throughout the lifetime of the CSS User Interfaces operating system process (even if potentially in an iconized state). It is therefore the tool which provides the functionality to quit the CSS User Interfaces.

Copyright per DIN 34

In the following, a description of the contents of the individual DBB subviews is presented:

Tool Bar

The **Set System Tree** button allows to select a system tree in the MDB. A selection window appears listing all available system trees.

The **Set CCU** button allows to select a CCU in the currently selected system tree. A selection window appears listing all available CCUs; it provides buttons allowing to restrict the CCUs listed to those that contain the CDU which is selected in the CDU subview, either in any or in the specific version.

If a CCU is selected, the CDU subview presents only the CDUs that are contained in the respective CCU.

Note that a CCU has to be selected in order to invoke the various CSS tools.

The **Reset CCU** button allows to deselect the CCU. If no CCU is selected, the CDU subview presents all CDUs in the selected system tree.

Note that as long as no CCU is selected, it is not possible to do any further work.

The **Rescan CDU** button allows to invalidate cached database information, causing the tool to re–initialize the display with updated values from the database.

The **QUIT** button quits the CSS session, ensuring that all tasks have been properly finished.

System Tree Subview

The system tree subview identifies the currently selected system tree. First part is the element configuration name (which is the first element in the system tree) (here "**EURECA**"), followed by the mission name (here "**DUMMY_MISSION**") and the system tree version (here "**V6**", see Figure 6–41).

CCU Subview

The CCU subview identifies the currently selected CCU. First comes the pathname of the node in the system tree where the CCU is located (here "**\EURECA\SIMULATOR**"), then the name of the CCU (here "**CSS_TEST**") and the CCU version (here "**V1.0.0**", see Figure 6–41).

CDU Subview

This subview shows the list of CDUs contained in the previously selected CCU. If no CCU is selected, all CDUs in the current system tree are listed.

In order to access simulation models, select the **Domain:CSS** button. The CDUs listed are filtered by domain CSS (the pop up menu of the item subview provides commands to create, delete and modify simulation models). Otherwise, to select an onboard item for referencing, select the **Domain:al** button. All CDUs regardless of their domain are listed (the pop up menu of the item subview provides commands for navigating in trees of virtual nodes).

Item Subview

The item subview shows the list of nodes under the currently selected CDU.

In a CDU of domain CSS, the nodes listed are simulation models. If the **Domain:CSS** button is selected, the pop up menu of this subview provides a variety of operations on simulation models depending on CDU resp. model status and ownership. In contrast to database based simulation models (i.e. simulation models the sources of which are kept in the database), filesystem based simulation models (i.e. simulation models the sources of which are kept in the filesystem) are marked with an icon. The sub-

Copyright per DIN 34

view allows multiple selection: Point to an item and click the middle mouse button to select it in addition to previously selected items.

In a CDU of a domain other than CSS, the nodes listed may be of any type. For CSS, only virtual nodes and onboard items (stimuli and measurements) are of interest. If the **Domain: all** button is selected, the pop up menu of this subview provides commands to navigate in trees of virtual nodes.

Info Subviews I and II

The contents of the Info Subviews I and II depend on the selections in the other subviews. Per default (if a CDU, but no item is selected), Info Subview I shows information about the currently selected CDU, such as status (development/revision/frozen), domain and owner, Info Subview II is empty.

If the **Domain: CSS** button is selected and a model is selected in the Item Subview, Info Subview I shows a number of buttons allowing to select a category of model attributes:

- **Info**
- **Simulation States**
- **Simulation Tables**

If the button **Info** is selected, Subview II shows information about the selected simulation model, i.e. model status, the permissions, the architectures for which the model is configured, the model size and modification date.

If one of the buttons **Simulation States** or **Simulation Tables** is selected in Info Subview I, the list of simulation states or the list of simulation tables belonging to the selected model is displayed in Info Subview II. The pop–up menu in Info Subview II gives access to the creation, modification and deletion of simulation tables resp. the deletion of simulation states.

If the **Domain: all** button is selected and an item is selected in the Item Subview, Subview II shows information about the selected item.

6.4.6.3 Accessing Simulation Models

Simulation models may be database or filesystem based. The source of a database based simulation model is kept in the database. If such a model is loaded, its references to onboard items that have become undefined or incompatible are detected. If appropriate, undefined onboard references may be adapted interactively, see section 6.4.7.8, otherwise they will be reset automatically. However, if the model is relatively large, the process of loading may be time consuming.

The source of a filesystem based simulation model is kept in the filesystem. The process of loading such a model is usually less time consuming than that of a database based model, specifically for large models there is a significant performance advantage. However, function block references don't work with these models, also references to onboard items that have become incompatible to the referencing interface items are not detected and reset when the model is loaded.

Database based simulation models may be exported into the file system and vice versa, filesystem based simulation models may be imported into the database.

Names of models, simulation tables and simulation states must comply with Ada identifier syntax, i.e.:

* the first character must be a letter
* any subsequent characters must be letters, digits, or the underscore ('_')
* two underscores cannot occur together, nor can a model name end with an underscore

Copyright per DIN 34

Names of simulation models must be unique in the scope of a CDU, with the exception that a database based and a filesystem based simulation model may have the same name. Names of simulation tables must be unique in the context of a specific simulation model, the same is true for simulation states.

Creating a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, deselect and choose **add model–>in database** or **add model–>in filesystem** from the pop–up menu, depending on whether you want to create a database based or a filesystem based simulation model.
- An input window appears: Type in the name of the new simulation model.
- The new simulation model appears in the Item Subview and is automatically selected.

Copying a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to copy and choose **copy** from the pop–up menu.
- Switch to another CDU, possibly in another system tree and/or in another CCU.
- In the Item Subview, deselect and choose **paste** from the pop–up menu.
- If a naming conflict occurs, an input window appears: Type in the name of the copied simulation model.
- The copied simulation model appears in the Item Subview and is automatically selected.

Renaming a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to rename and choose **rename** from the pop–up menu.
- An input window appears: Type in the new name of the simulation model.
- The simulation model’s new name appears in the Item Subview, the model stays selected.

Deleting simulation models

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model(s) you want to delete and choose **delete** from the pop–up menu.
- The selected simulation model(s) disappear(s) from the Item Subview.

Exporting a simulation model from the database into the file system

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the database based simulation model you want to export and choose **export** from the pop–up menu.

Copyright per DIN 34

- If a naming conflict with another filesystem based simulation model occurs, an input window appears: Type in the name of the exported filesystem based simulation model.
- The new filesystem based simulation model appears in the Item Subview and is automatically selected.

Importing a simulation model from the filesystem into the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the filesystem based simulation model you want to import and choose **import** from the pop–up menu.
- If a naming conflict with another database based simulation model occurs, an input window appears: Type in the name of the imported database based simulation model.
- The new database based simulation model appears in the Item Subview and is automatically selected.

Not all parts of a configured database based simulation model are stored in the database. The model's executable images (simulator kernels), the simulation states, the mapping tables and the adaptation system configuration files for the various architectures are kept under the user's own account in the UNIX filesystem.

If simulation models shall be part of a data contents transfer between MDB instances via the MDA import/export mechanism, they must be stored in the database completely. The **pack** command allows to store the data normally kept in the filesystem into the database. The corresponding files are deleted automatically. A packed simulation model is ready for an export.

Packing configured database based simulation models in the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, **select** the model(s) and choose **pack** from the pop up menu.

It is not possible to do any further work with a packed simulation model unless it is unpacked; this has to be done after an import has been taken place. The **unpack** command allows to revert the effect of the pack operation, i.e. to reestablish the corresponding data in the filesystem.

Unpacking configured database based simulation models in the database

- **Select** the **Domain: CSS** button.
- In the Item Subview, **select** the model(s) and choose **unpack** from the pop up menu.

- *Note that the pack and unpack operations always comprise all architectures the model has been configured for.*

CSS provides a number of tools that can be opened on a simulation model from within the Database Browser: the Composite Interface Inspector, the Composite Inspector, the Composite Interface Editor, the Composite Editor, the Hierarchy Browser, the Documentation Tool and the Simulation Controller.

Opening a tool on a simulation model

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to open a tool on and choose one of the commands **inspect–>interface**, **inspect–>implementation**, **edit–>interface**, **edit–>implementation**, **browse hierarchy**, **print** or **execute** from the pop–up menu.

Copyright per DIN 34

- The tool opens in a separate window.

Simulation tables and simulation states are attributes to simulation models.

Creating a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to add a simulation table to.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, deselect and choose **add table** from the pop up menu.
- An input window appears: Type in the name of the new simulation table.
- The new simulation table appears in the Info Subview II and is automatically selected.

Copying a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to copy a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to copy and choose **copy** from the pop up menu.
- Select the simulation model you want to add the copied table to, possibly in another system tree and/or in another CCU and/or in another CDU.
- In the Info Subview II, deselect and choose **paste** from the pop–up menu.
- If a naming conflict occurs, an input window appears: Type in the name of the copied simulation table.
- The copied simulation table appears in the Info Subview II and is automatically selected.

Renaming a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to rename a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to rename and choose **rename** from the pop up menu.
- An input window appears: Type in the new name of the simulation table.
- The simulation table’s new name appears in the Info Subview II, the table stays selected.

Deleting a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to delete a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.

Copyright per DIN 34

- In the Info Subview II, select the simulation table you want to delete and choose **delete** from the pop up menu.
- The selected simulation table disappears from the Info Subview II.

CSS provides two tools that can be opened on a simulation table from within the Database Browser: the Simulation Table Inspector and the Simulation Table Editor.

Opening a tool on a simulation table

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to open a tool on a simulation table from.
- In the Info Subview I, select the **Simulation Tables** button.
- In the Info Subview II, select the simulation table you want to open a tool on and choose **inspect** or **edit** from the pop up menu.
- The tool opens in a separate window.

Deleting a simulation state

- **Select** the **Domain: CSS** button.
- In the Item Subview, select the simulation model you want to delete a simulation state from.
- In the Info Subview I, select the **Simulation States** button.
- In the Info Subview II, select the simulation state you want to delete and choose **delete** from the pop up menu.
- The selected simulation state disappears from the Info Subview II.

6.4.6.4 Selecting an Onboard Item

In contrast to simulation models, onboard items (i.e. stimuli and measurements) are usually not located as subnodes directly under a CDU. Instead, possibly together with end items of various types, they may be organized in a tree structure of virtual nodes. To select an onboard item, it is necessary to navigate in this tree.

Make sure that a CCU is selected and press the **Domain: all** button in the DBB window. The CDU Subview shows the list of all CDUs contained in the current CCU. Select one of the CDUs. The list of subnodes is displayed in the Item Subview. If an item is selected, Info Subview II shows information about the selected item, at least whether it is a virtual node, a simulation model or an unknown end item. If the item could be identified as an onboard item, it is stated whether the item is a stimulus or a measurement and further information is provided, i.e. the corresponding CSS data type and state codes or engineering unit, if appropriate.

If a virtual node is selected, a pop up menu command **expand** allows to view the subnodes unter that virtual node. The pathname (relative to the selected CDU) of the current virtual node is presented on top of the Item Subview; the Item Subview provides the subnodes under that virtual node. If no item is selected in the Item Subview, the pop up menu command **collapse** allows to step back to the next higher level in the tree of virtual nodes.

Copyright per DIN 34

6.4.7 MDE User Interface

6.4.7.1 Composite Editor

6.4.7.1.1 Basics

The Composite Editor is activated in a separate MDE window. It is the basic window for an edit session on a particular simulation model. Atomic Editors and Icon Editors can be started from within the Composite Editor. Only one Composite Editor may be opened on a specific model.

User Interface

Figure Figure 6–42 shows the user interface of the composite editor. The tool provides a composite edit view (*edit view*) that allows interactive model editing using MDE–GL. It provides vertical and horizontal scrolling functions. Additionally to the normal edit mode the view can be run in overview mode. A bar of iconized buttons allows the user to influence the behaviour of the composite edit view (e.g. to specify the MDE–GL syntax element that should be created next, to switch from edit mode to overview mode and vice versa). A read only text view is used to provide textual user feedback (*message subview*).

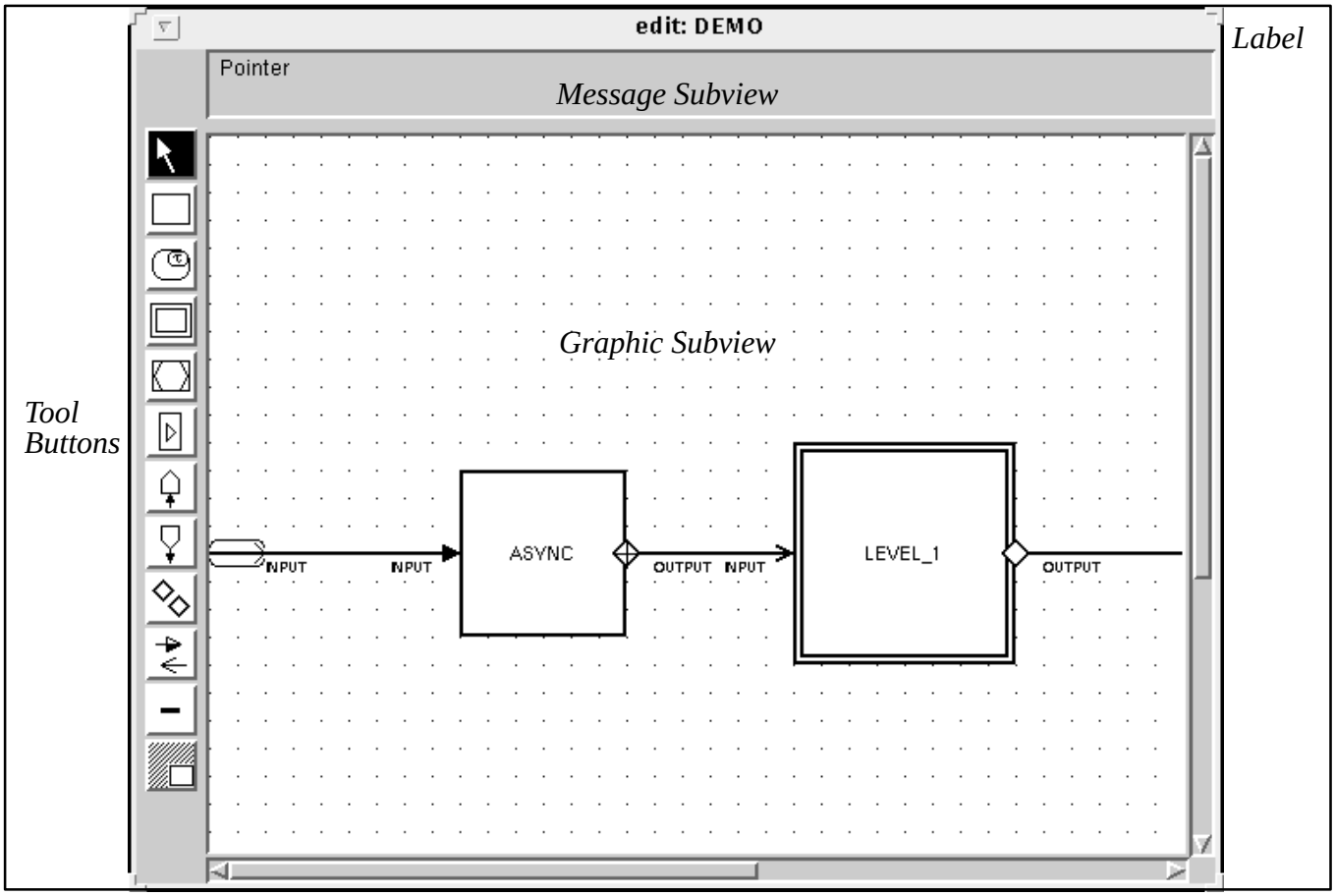


Figure 6–42 : *The Composite Editor User Interface*

Copyright per DIN 34

6.4.7.1.2 The Composite Editor's Components

6.4.7.1.2.1 The Label

The label of the Composite Editor shows the pathname of the composite function block that's inside view is currently displayed in the graphic subview. It is called the current level in the hierarchy of composite function blocks.

When you open a Composite Editor the model itself is initially set to the current level.

6.4.7.1.2.2 The Graphic Subview

The scrollable graphic subview shows a part of the inside view of the composite function block that's pathname is shown in the label. Note that the inside view of the just created composite function block is empty.

Move the cursor into the scroll bar area. Press the left button and hold. The bar moves to the cursor location and then tracks the cursor until the left button is released. The displayed document jumps to the appropriate location.

The scrollbars provide additional buttons. Clicking on them allows scrolling step by step in the related direction.

6.4.7.1.2.3 The Tool Buttons

The operation initiated by pressing the middle mouse button inside the graphic subview depends on which of the tool buttons is currently activated. The active tool button appears highlighted. Select different buttons by moving the cursor over the button and clicking the left mouse button. Note that only one tool button can be active at a time, i.e. the activation of a particular button automatically deactivates the previously active button.

Note that clicking the left button inside the graphic subview always sets the pointer button to be the active button.

Editorial Note:
Since the Frame Synchronization Point has been added only recently, most screen snapshots of the model editor window in this chapter have not been updated to include the corresponding icon.

Copyright per DIN 34

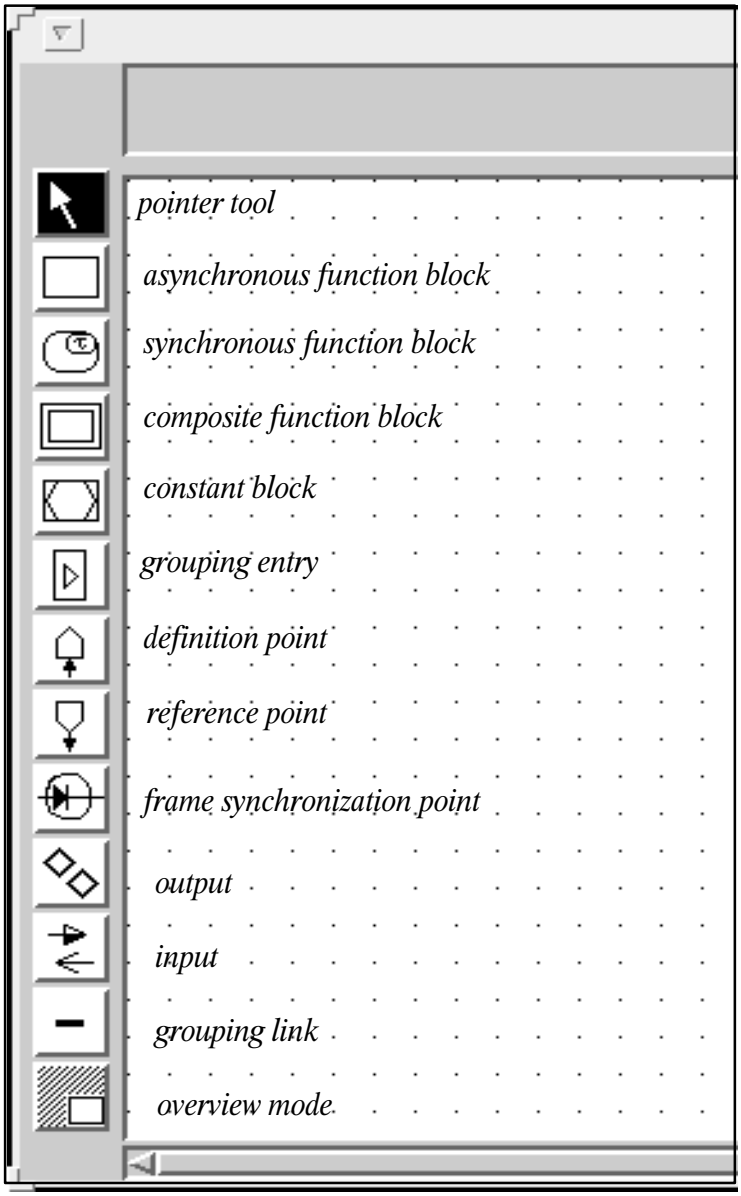


Figure 6–43 : *The tool buttons of the Composite Editor*

6.4.7.1.2.4 The Message Subview

While editing the message subwindow is used to provide feedback. For example if you try to perform an illegal action you will be informed by a message indicating the mistake.

6.4.7.1.2.5 The Overview Mode

In general the editor window will be not large enough to display the entire graphical document (i.e. the inside view of the actual displayed composite function block).

The *overview mode* allows you to see the whole document at a time. It also provides a convenient way to scroll to a particular location. To enable overview mode, activate the overview

Copyright per DIN 34

button.
The actual visible part of the document is indicated by the white underlaid rectangle while the other parts are grey underlaid.

You can change the visible part by scrolling horizontal or vertical using the particular scroll bar.

A more convenient way is to move the cursor inside the white underlaid area and press the left button. Note that the cursor shape changes to that of a hand. Holding the button pressed move the cursor around.

To go back to *edit mode* press one of the tool buttons (except the overview button) or select **edit** from the pop–up menu also available by pressing the right mouse button.

The previously white underlaid part of the document becomes the visible part in the graphic subview.

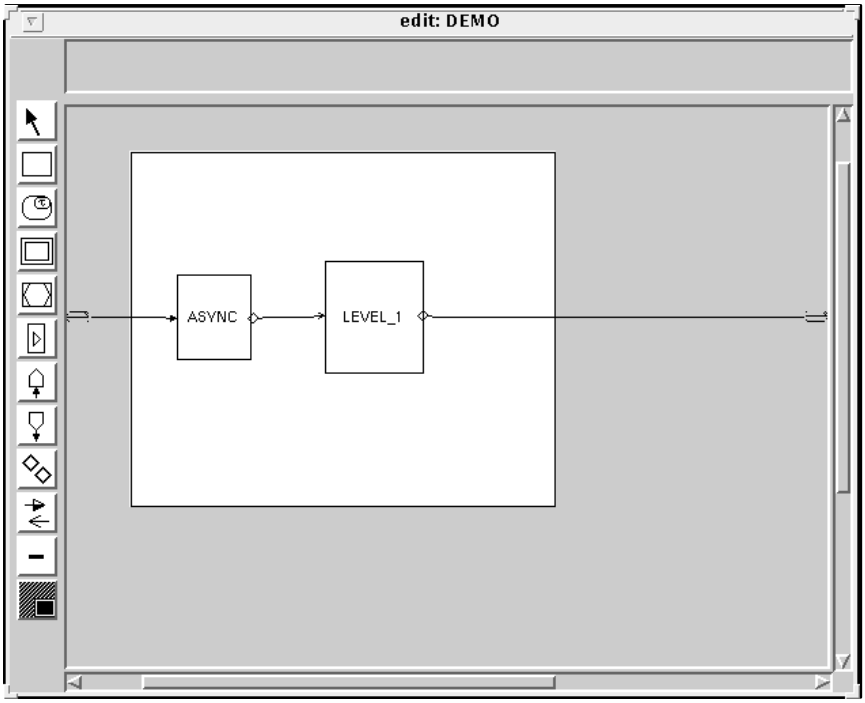


Figure 6–44 : *Scrolling in overview mode*

6.4.7.1.3 Selection Sensitive Menus

There are a number of different pop–up menus. The menu that actually appears when you press the right mouse button depends on whether or not there are selected objects and what types of objects are selected. Even the numbers of selected objects can affect the appropriate menu.

Copyright per DIN 34

6.4.7.1.4 The Composite Editor's Basic Menu

Move the cursor inside the graphic subview and press the right mouse button. You obtain the Composite Editor's basic menu (i.e the menu that appears when there are no selected objects).

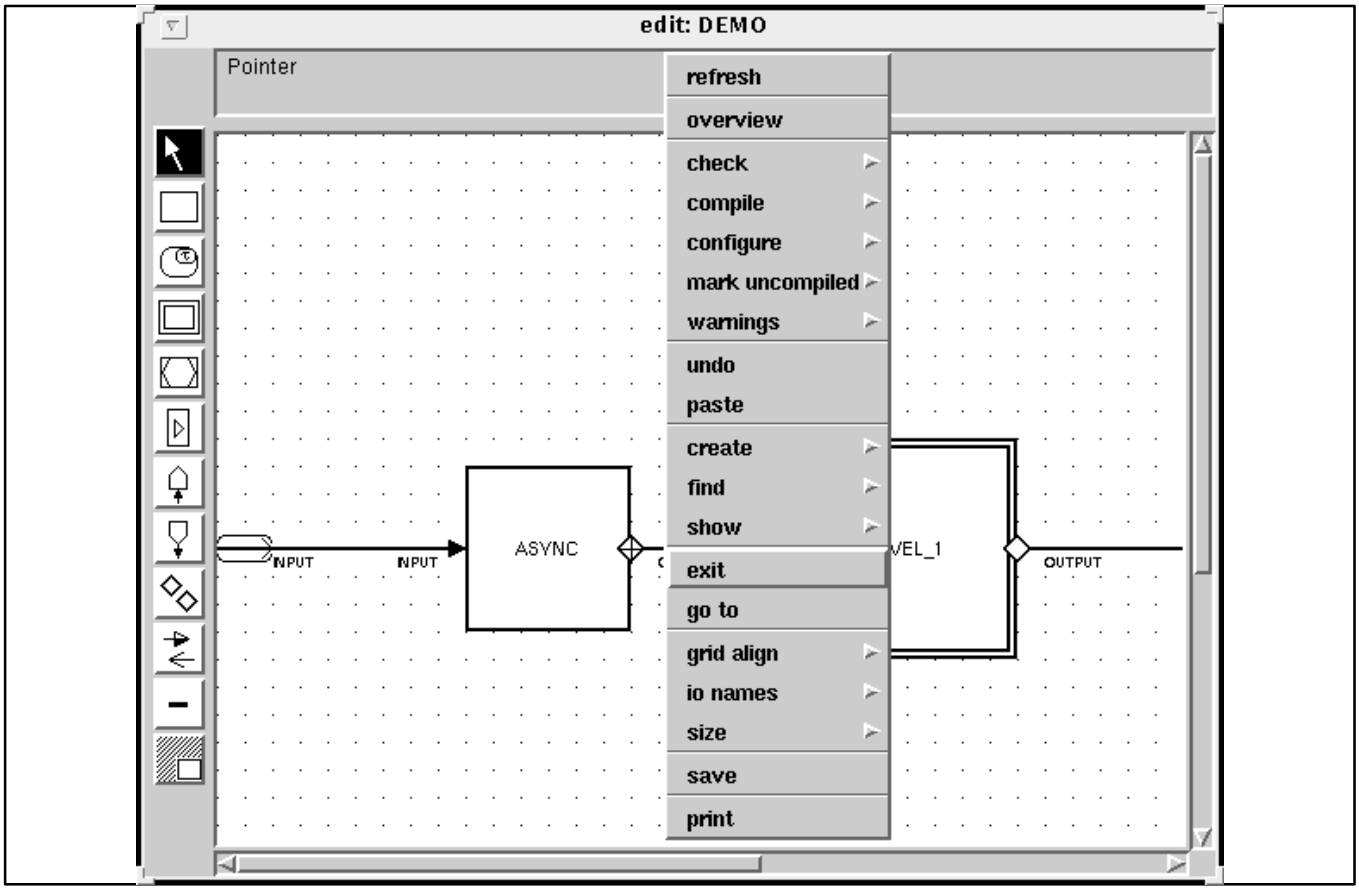


Figure 6–45 : *The Composite Editors basic menu*

6.4.7.1.5 Creating Block Objects, Grouping Entries, Global Symbols and Frame Synchronization Points

Remember that the actions invoked by pressing the middle mouse button depend on which tool button is activated. Activate the tool button labeled with an asynchronous function block by moving the cursor over the tool button and clicking the left mouse button. Now you can create one or more asynchronous function blocks.

To do so, move the cursor inside the graphic subview and click the middle button. A black rectangular outline appears on the screen. The size of this rectangle indicates the minimum size of the specific kind of object, in this case the minimum size of the graphical symbol for an asynchronous function block. Move the cursor around without pressing a button.

Note that the black rectangular outline tracks the cursor. Move it to the location you want to draw the asynchronous function block and press the middle button. If you hold the middle

Copyright per DIN 34

button and move the cursor you can resize the just created function block. Release the mouse button, an asynchronous function block appears on the screen.

The last created object is automatically selected. In this case this is indicated by black selection marks. Create another asynchronous function block. See that the first function block is automatically deselected.

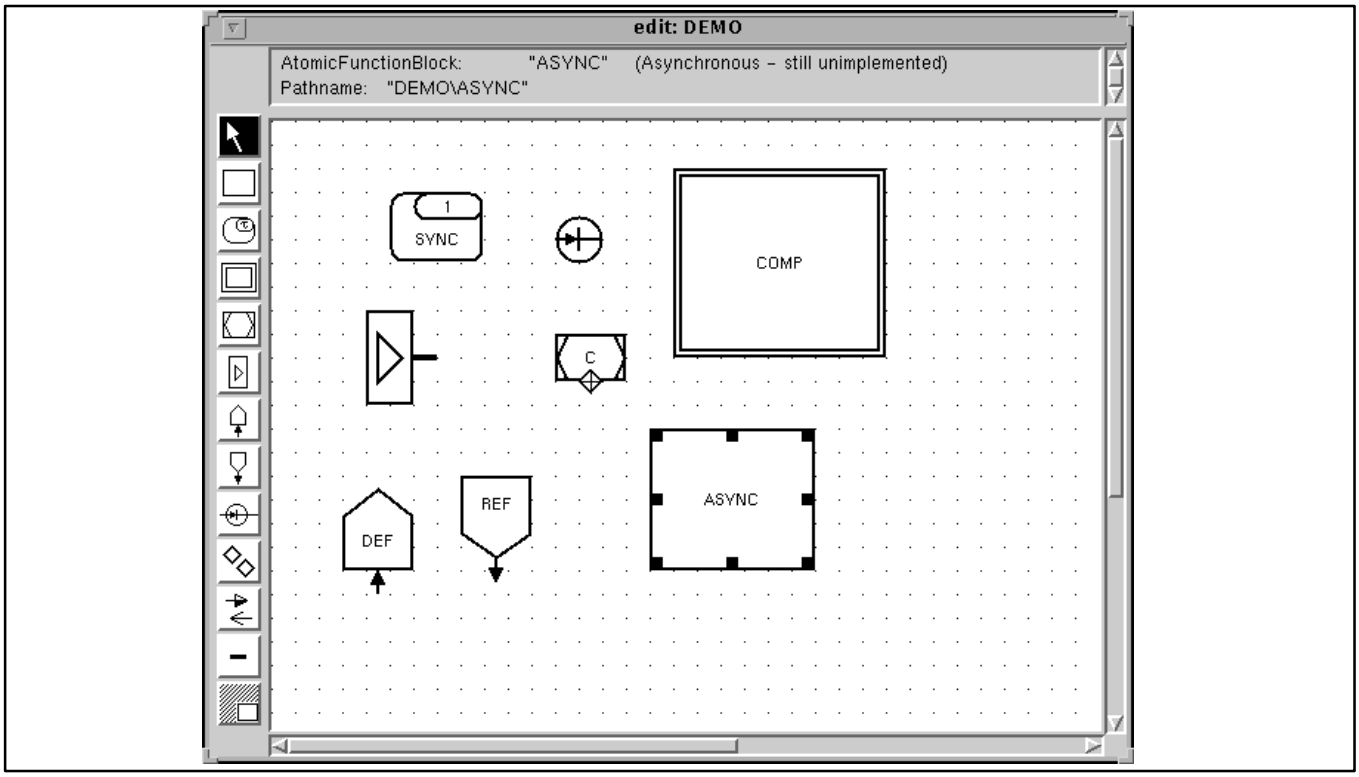


Figure 6–46 : *A Composite Editor showing some block objects. The last created asynchronous function block is selected.*

The same way you can create synchronous and composite function blocks as well as constant blocks and grouping entries. For each of these objects there is an appropriate tool button. Note that each constant block is created including an output symbol.

Definition point, reference point and frame synchronization point are created in a similar way. The only difference is that you can't specify a size by moving the pressed middle button. Click the middle button to get the black rectangular outline. Move it to the desired location and simply click the middle button again.

Another way to initiate creation of one of these objects is to choose the basic menu command **create**. To do so, move the cursor inside the graphic subview to an empty location and click the left mouse button to deselect all objects. Then press the left mouse button to get the basic menu. Drag the cursor up to **create**. A submenu appears listing the different objects you can create.

Choose the submenu item **composite** to create a composite function block. The black frame appears on the screen inviting you to determine location and size of the graphical symbol. Simultaneously the tool button labeled with a composite function block is set to be active.

Copyright per DIN 34

6.4.7.1.6 Creating Outputs, Inputs, Grouping Links

To create an output, input or a grouping link, activate the appropriate tool button by moving the cursor over the button and clicking the left mouse button.

Each i/o–item you create has to be attached to an asynchronous, synchronous, composite function block or a grouping entry (grouping links can only become attached to composite function blocks).

You specify the target function block by moving the cursor over its symbol. Then click the middle button. A black frame appears that can be moved on the function block's border lines by moving the cursor around. Move the frame to a place you want the i/o–item to appear. Click the middle button once again to finish creation.

If the cursor is not positioned over a particular function block, then the function block nearest to the cursor position is assumed to be the destination function block.

The graphic subview always represents the inside view of the composite function block you are currently editing. You can also assign i/o–items to this composite. Move the cursor near the border of the inside view but don't leave the graphic subview (maybe you first need to scroll to reach the border of the inside view). This has the effect that the currently edited composite is supposed to be the target function block. Click the middle button. Move the appearing black frame to a suitable position and click the middle button again.

Note that the last created i/o–item is automatically the selected object. The selection of an input or grouping link is indicated by a black dot at the point of the arrow or line.

Assign an input to an atomic function block.

Note that its default activation mechanism is always.

Copyright per DIN 34

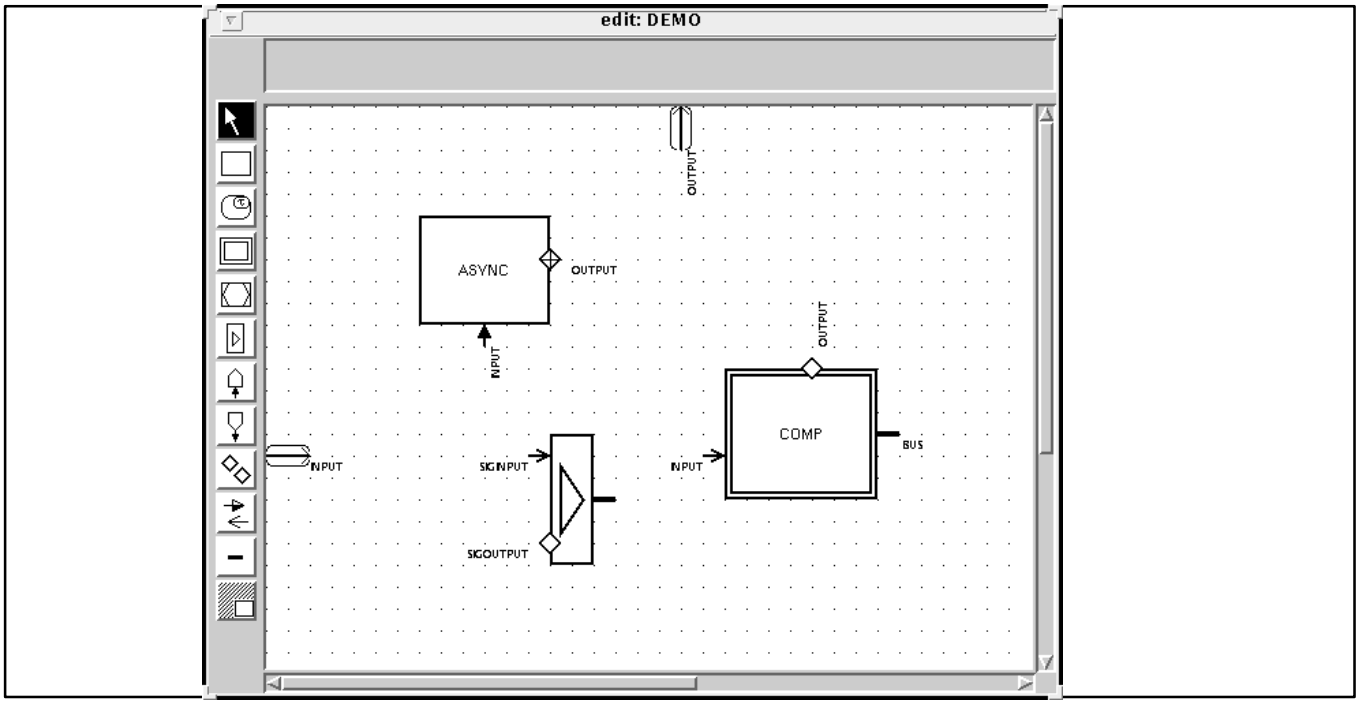


Figure 6–47 : *A Composite Editor showing some I/O items attached to block objects. The last created output is selected.*

Create a number of i/o–items to a function block. Note that you can't specify a location that another i/o–item occupies. Note also that the maximum number of i/o–items depends on the size of the function block's graphical symbol. If this limit is reached and you want to create a further i/o–item, the operation will become aborted. You will be informed by a message in the message subwindow.

6.4.7.1.7 Selecting Objects

There are two different ways of selection. Exclusive selection means that the object you want to select should be the only selected object (i.e. eventually previously selected objects should automatically become deselected). Additional selection means that an object should become selected in addition to other selected objects.

Exclusive selection is performed by moving the cursor over the graphical symbol and clicking the left button; additional selection is performed similarly by use of the middle button.

Note that additional selection is restricted to objects of the same type. So you can not select an output in addition to function blocks. If you try, the output will become selected while the function blocks are automatically deselected.

The selection of a function block involves the selection of the related i/o – items, connection lines and i/o – item names (not visibly selected) in the context of moving and deleting function blocks.

6.4.7.1.8 Deselecting Objects

To deselect all selected objects at once simply move the cursor inside the graphic subview to an empty location and click the left button.

Copyright per DIN 34

To deselect a specific object make sure that the pointer tool button is activated. Move the cursor over the selected graphic symbol and click the middle button.

6.4.7.1.9 Renaming Objects

Select the object you want to rename and press the right mouse button. Execute **rename**. A text field appears asking you for the new name. Type in the new name.

Note that there is no dedicated name for a frame synchronization point.

6.4.7.1.10 Resizing Block Objects

Select the object you want to resize. Move the cursor over one of the black selection marks. Press the left button. The cursor changes shape and a rectangular outline appears on the screen. Move the cursor around. Correspondingly the rectangle changes its size indicating a possible size for the graphical symbol. If you are satisfied, release the button. The resized block appears on the screen.

To change height and width of symbol at once, choose a selection mark that lies at one of the corners. The cursor changes shape to look like the appropriate corner of a rectangle.

This operation always maintains the minimum size of the function block.

Note that the minimum size of a function block depends on the number of i/o–items.

All block objects except definition point, reference point and frame synchronization point can be resized.

6.4.7.1.11 Rotating Block Objects

All block objects except definition point and reference point can be rotated. Select the object you want to rotate and execute the menu command **rotate**. Note that the function block rotates by 90 degrees counterclockwise.

6.4.7.1.12 Moving Block Objects

You can move a block object to another place inside the graphic subview. To do this you have to select the object first. Then move the cursor over the block (but not over a selection mark) and press the left button. A black frame indicating the borders of the block appears on the screen.

Move the cursor around holding the button down. See that the black frame tracks the cursor. Release the button if you are satisfied with the new position. The block appears at the new position.

You can also move several blocks at once. Select a number of blocks and position the cursor over one of these (but not over a selection mark). Press the left button and move the cursor

Copyright per DIN 34

around. For all objects one large frame tracks the cursor indicating its borders. Finish moving by releasing the left button. All selected blocks appear at their new position.

6.4.7.1.13 Moving I/O–Items

You can move a specific i/o–item to another place on the border lines of its associated function block. To do this you have to select the object first. Then move the cursor over the i/o–item and press the left button. A frame indicating the position of the i/o–item appears on the screen.

Holding the button pressed, move the cursor around. See that the frame tracks the cursor as long as a position is not occupied by another i/o–item. Release the button if you are satisfied with the new position. The i/o–item appears at the new position.

6.4.7.1.14 Placing Block Objects into the foreground resp. background

Block objects are allowed to overlap.

Create a number of these objects and move them so that they overlap partially. Select an object that is partially hidden. Choose the menu item **arrange**. A submenu appears listing two commands: **to front** and **to back**. Execute **to front**. Note that the selected block is placed into the foreground. Now execute **to back**. The selected object is placed into the background.

These operations can also be applied to a group of block objects.

6.4.7.1.15 Removing Block Objects

Select the object(s) you want to remove. Execute the menu command **cut**. The objects disappear.

The basic menu commands **undo** allows you to get back the last removed blocks.

6.4.7.1.16 Removing I/O–Items

Select the i/o–items(s) you want to remove. These may belong to different function blocks. Execute the menu command **cut**. The i/o–items disappear.

The basic menu command **undo** allows you to get back the last removed i/o–items.

6.4.7.1.17 Copying Block Objects

You can copy one object as well as several objects at once. Select the object(s) you want to copy. Execute the menu command **copy–>normal**. Then execute **paste**. Move the cursor around. Correspondingly the black rectangle (containing one or more copied objects) changes its position inside the graphic subview. Move the objects to a position where they don’t overlap other objects and click the middle button. The copies appear on the screen.

6.4.7.1.18 Copying Block Objects per reference

You can copy one object as well as several objects per reference from models which are in a CDU configured in your currently used CCU.

Copyright per DIN 34

Copying block objects per reference

- The model editor window is open.
- Move the mouse cursor into the DBB window and **select** the CDU containing the model you want to reference in the CDU subview. The list of models appears in the models subview.
- **Click** on the desired model.
- Press the right mouse button and select **inspect –> implementation** from the pop–up menu. The inspector window is opened and shows the model in overview mode.
- Press the right mouse button and select **inspect** from the pop–up menu.
- Navigate to the desired part of the model.
- Select one or more block objects.
- Press the right mouse button and select **copy –> reference** from the pop–up menu.
- Move the cursor back to the editor window, press the right mouse button and select **paste**. A rectangle appears.
- Move the rectangle to the desired position in the model, then press the middle mouse button. The copied block objects appear with a grey underlay.
- Move the cursor back to the inspector window and select **Quit** from the window menu.
- Select the initial CDU in the CDU subview.

It is possible to set references to models which have the DB status develoment. Note the following restriction:

- as soon as the referenced model is stored (not necessarily changed !) all references are invalid i.e. next time the model is opened all (now invalid) references will be resolved, which can be a very time consuming procedure.

6.4.7.1.19 Changing the Activation Characteristics of Inputs

When you assign a new input to an atomic function block, its default activation characteristic is *always*. To specify another activation characteristic, select the input(s) and choose the menu item **activation**. A submenu appears listing the three different activation characteristics **always**, **on change** and **never**. Choose the desired characteristic. The graphical representation of the selected inputs will become updated.

*Note that the activation mechanism n **never** should only be used to connect constant blocks to asynchronous function blocks or as a timing control mechanism to avoid asynchronous loops.*
*Changing values transmitted via a connection with **never** characteristics will be automatically delayed for one time frame to avoid asynchronous loops.*

6.4.7.1.20 Stepping Through Composite Hierarchies

6.4.7.1.20.1 Stepping Into a Lower Level Composite Function Block

Select one composite function block and execute the menu command **enter**. The graphic area now shows the inside view of the particular composite function block. Note that the editor

Copyright per DIN 34

initially changes to overview mode. Note also that the label now shows the full pathname of the composite function block you stepped in. Go back to edit mode by choosing the menu command **edit** or by clicking the left mouse button over one of the tool buttons.

6.4.7.1.20.2 Stepping Back to the Next Higher Level

To go back to the parent composite function block, perform the following actions:
Deselect all objects to get the basic menu. Drag the cursor to **exit** and release the left mouse button. Note that the label changes the displayed name to that of the parent composite function block. Simultaneously the graphic subview displays the contents of the parent composite function block.

Note that there is one composite function block selected. This is the composite you just left.

6.4.7.1.20.3 The Tree Browser

Execute the basic menu command **go to**. A Tree Browser running in a separate window appears on the screen. It shows the composite hierarchy of the edited model.

Note : Since the collection of the data required for the Tree Browser display, especially for a large model, may take quite some time, the user is prompted for confirmation before the Tree Browser is actually started.

The Tree Browser allows to step to an arbitrary composite function block. Inside the Tree Browser, click the left mouse button on the name of the function block you want to go to. In the Composite Editor, execute **go to** once again. Correspondingly the Composite Editor changes the actual edited composite function block and selects the function block that is selected in the Tree Browser.

For more details about the Tree Browser user interface see the related chapter.

6.4.7.1.21 Connecting Objects

Select the objects you want to connect and press the right mouse button. If the selected objects are allowed to become connected, the menu provides a command **connect**. Execute **connect**. If ok, the items are connected graphically.

Note that the MDE prevents you from connecting type incompatible objects.

Copyright per DIN 34

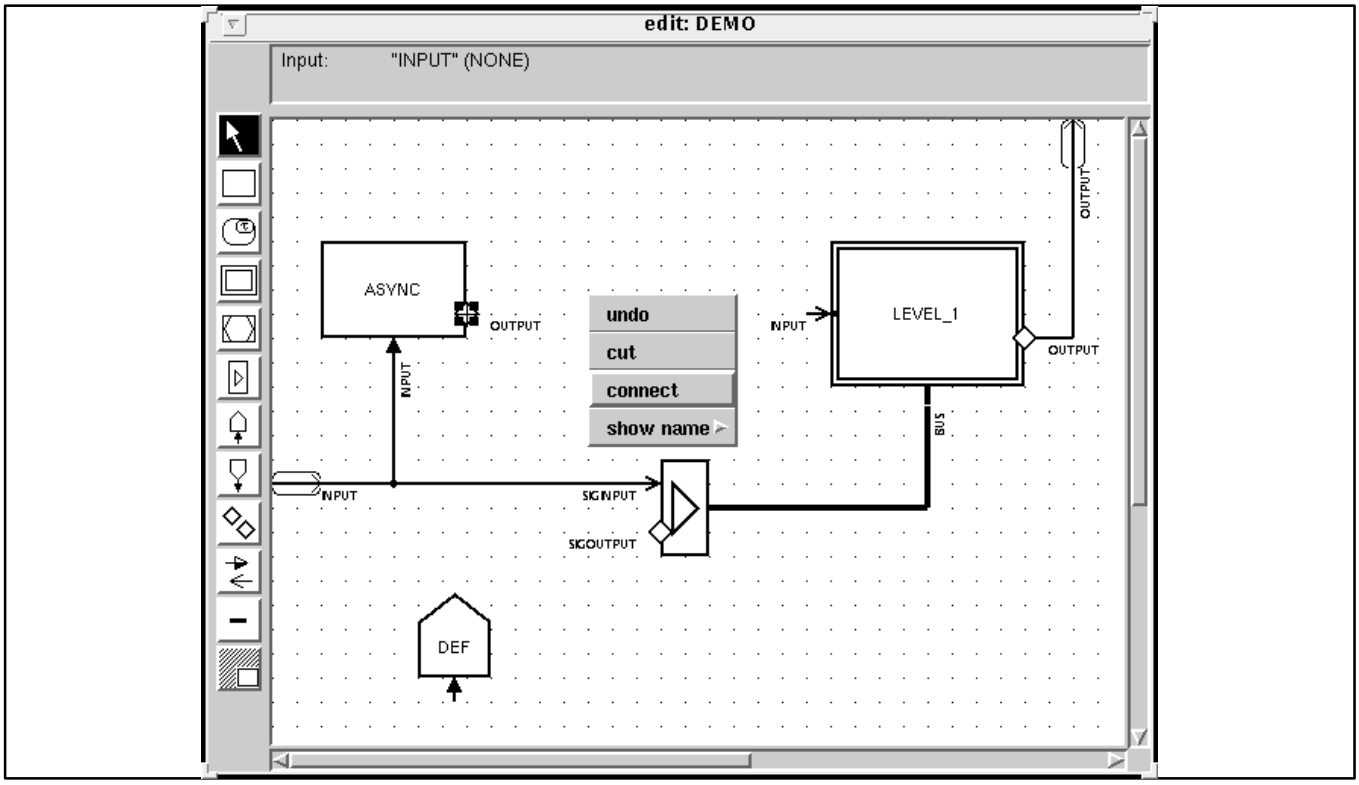


Figure 6–48 : *The output of function block ASYNC will be connected with the INPUT of COMP.*

6.4.7.1.22 Connecting top level I/Os to onboard items

If the model is intended to be used for H/W in the loop simulations, the connection between the simulator and the external H/W has to be defined by referencing onboard items specified in the database from the model's interface items (top level I/Os).
Following precondition has to be fulfilled: at least one CDU containing onboard items is configured in the CCU you selected previously.

- *The user needs at least two CDUs to get access to the model and to the onboard items.*
- *We recommend to create the onboard items in the database first and then to connect the onboard items with the CSS top level I/Os.*

Copyright per DIN 34

Connecting top/level I/Os to onboard items

- The CSS model editor window is open.
- Top level inputs/outputs are created.
- *Note that top level I/Os can be undefined (no type setting, no initial value)*
- Move the mouse pointer into the DBB window and press the **all** button. (see Figure 6–49) A list of CDUs appears in the CDU subview.
- Select the desired CDU in the CDU subview (on the left side) by clicking on the name with the left mouse button.
- Navigate to the desired end item by selecting a virtual node in the right subwindow and then pressing the right mouse button and execute **expand**.
- *To navigate from an end item to a higher node, deselect the item, then press the right mouse button and select **collapse**.*
- Move the mouse pointer back to the model editor window.
- Select the I/O item by clicking on it with the left mouse button, then press the right mouse button and select **variable –> set onboard reference** from the pop–up menu.
- For top level inputs only: Press the right mouse button and select **variable –> initial value ...** from the pop–up menu. Type/select the initial value (a default initial value is set automatically).
- *Repeat the last four steps until all top level I/Os are connected..*
- Deselect the I/O then press the right mouse button and select **save** from the pop–up menu.

Figure 6–49 : *The stimulus LONG_REAL is selected in the database*

It is also possible to set onboard references from a file in ASCII format. The command **O/B References–>Set From File** is available if the Composite Editor is located on top level (i.e. it shows the MDE–GL implementation of the top level composite function block) and no item is selected. It allows to read such a file and to set the references of model interface items to onboard items as specified there.

Copyright per DIN 34

The required format of such an ASCII file is described here:

- lines starting with # are considered as comments and are thus ignored
- each entry is stored in a separate line
- an entry consists of the following elements, all separated by one or multiple <space> and/or <tab>, respectively
- model interface item name
- the pathname of the CDU containing the referenced onboard item
- the pathname of the referenced onboard item relative to the CDU
- optionally a comment, i.e. some explaining arbitrary text

Entries with a model interface item name that does not refer to an actual model interface item are ignored, undefined onboard references (i.e. references to onboard items that cannot be accessed resp. are not visible in the current CCU scope) may be adapted interactively, see section 6.4.7.8.

The pop–up menu command **Variable–>Reset Onboard Reference** provides the possibility to reset (i.e. delete or clear) the association between the selected top level interface item and onboard item.

If the Composite Editor is located on top level, two commands are available allowing to reset multiple onboard references at once. The command **O/B References–>Reset All** is available if no item is selected and may be used to reset all of the model's references to onboard items. The command **O/B References–>Reset** is available if a grouping link at the model (i.e. top level) interface is selected. It allows to reset the onboard references of the model interface items belonging to the selected logical grouping.

It is possible to generate a listing of the model interface items and the referenced onboard items in an ASCII file. The format is the same as described above, so the file may later be used to set onboard references from. Two pop–up menu commands are provided which are available if the Composite Editor is located on top level. If no item is selected, the command **O/B References–>Write All To File** allows to list the complete model interface. The command **O/B References–>Write To File** is available if a grouping link at the model (i.e. top level) interface is selected. It allows to list those model interface items belonging to the selected logical grouping. In both cases the listing comprises also those model interface items that have no onboard reference. Each entry has a comment that gives information about the type of the model interface item and of the referenced onboard item, if there is one.

6.4.7.1.23 Selecting a Connection

Point with the cursor to the connection you want to select. Click the left button. Note that the end of each of these lines is indicated by a black dot. This informs you that the connection is selected.

Note that there can only be one connection selected at a time. Additional selection is not provided.

Copyright per DIN 34

6.4.7.1.24 Deselecting a Connection

To deselect a connection simply move the cursor inside the graphic subview to an empty location and click the left button.

Another way is to point the cursor to the selected connection and click the middle button. But before doing so you have to make sure that the pointer button is activated.

6.4.7.1.25 Removing a Connection

Select the connection you want to remove. Execute the menu command **cut**. The connection disappears. The affected i/o–items are in unconnected state.

The basic menu command **undo** allows you to get back the last removed connection.

6.4.7.1.26 Disconnecting an i/o–item

You know that one i/o–item can be connected to a group of i/o–items. For example think of an atomic output connected to several inputs. If you cut a connection all affected i/o–items become disconnected. So cutting is not the right way if you want to disconnect only one of the inputs.

To do so, select the appropriate i/o–item and press the left mouse button. Execute disconnect. See that only the connection lines from the selected i/o–item up to the next junction disappear.

6.4.7.1.27 Moving Connection Lines

When you connect i/o–items the MDE creates connection lines by default.

You can move each of these lines. Horizontal lines can be moved up and down, vertical lines can be moved to the right resp. to the left. Select a connection to see the end point of each line indicated by a black dot.

Point the cursor to the particular line that you want to move. Press the left button. The line starts blinking. Move the cursor around. Note that, as the cursor moves, the line follows accordingly. Probably one or two additional lines are used to connect the moved line to the rest of the connection. This rubber band function tells you the currently suggested solution to substitute the original line.

When you are satisfied, release the left button. The last provided solution is inserted into the connection substituting the original line.

6.4.7.1.28 Splitting Connection Lines

It may be possible that you want to move only a part of a longer connection line. To do so, you have to split this line into two separate lines. Select the connection and press the left

Copyright per DIN 34

mouse button and choose the item **split**. Note that the message subview shows the following text: *Please click any button at the point where you want to split*.

So move the cursor over the particular line that you want to divide into two lines. Click the left button. The connection stays selected showing the new line's end points.

The basic menu command **undo** allows to get back to the last unsplit connection line.

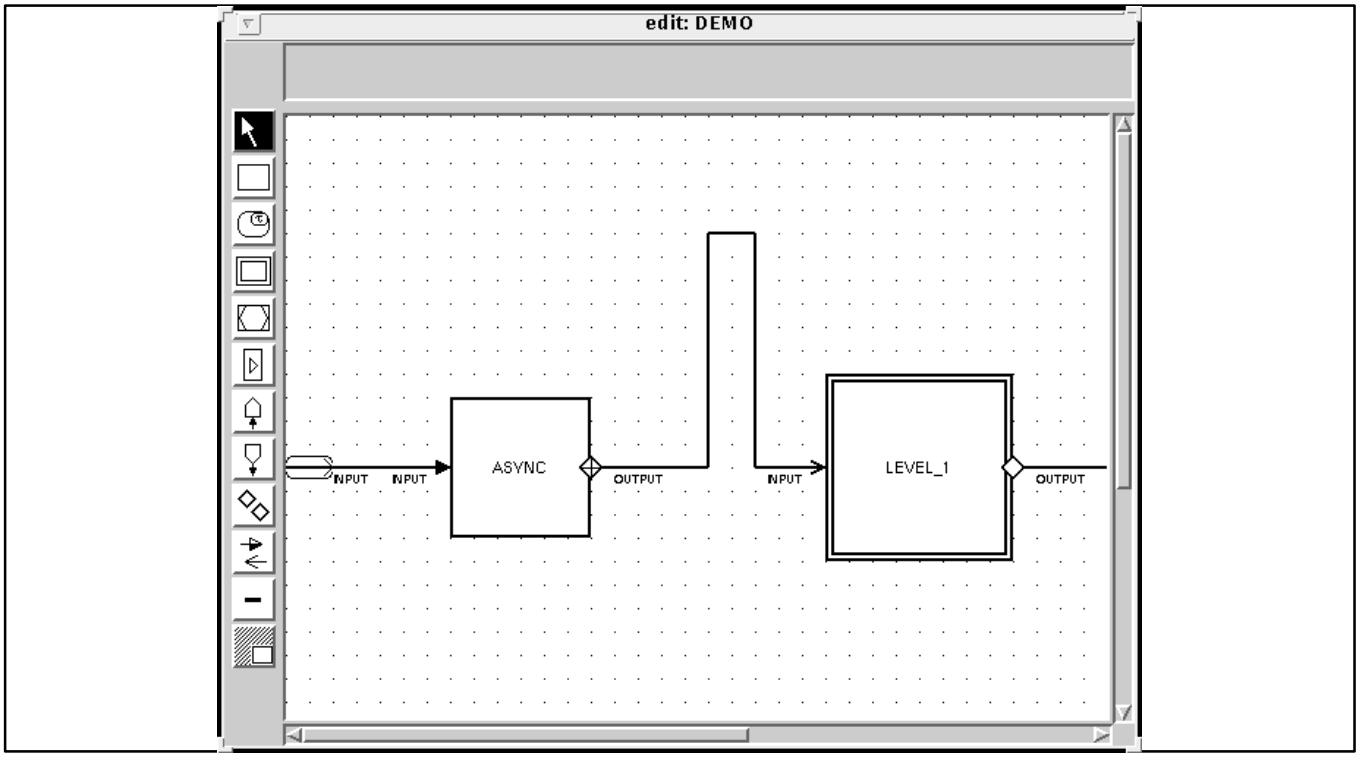


Figure 6–50 : *Moving a splitted connection line*

6.4.7.1.29 Manipulating connected objects

Almost every operation appropriate to block objects and i/o–items can be performed when these objects are connected, too. This comprises moving, resizing, cutting, copying and pasting. The affected connections will become updated automatically.

6.4.7.1.30 Logical Groupings

A logical grouping is a bidirectional bus that combines an arbitrary number of signals. Grouping entries provide the interfaces to a logical grouping. The interfaces to the signals contained in the logical grouping are the inputs and outputs of the grouping entries.

A completely defined signal has one input to a logical grouping and a number of outputs from the logical grouping. Each of these items representing the signal has the same name (the signal's name). Note that each signal can be interfaced only once at a particular grouping entry.

Create two grouping entries and select the associated grouping links. Execute **connect**. Create a composite function block and assign a grouping link to it. Select the grouping link

Copyright per DIN 34

and one of the grouping entries' grouping link and execute **connect** again. Step inside the composite function block and note that the grouping link is visible inside, too. Now create another grouping entry and connect its grouping link to that of the composite function block seen from inside.

Until now, the logical grouping does not contain any signals. Assign inputs and outputs to the grouping entries. Note that each of these items initially gets a name that is unique in the entire logical grouping. Each of these items represents a signal that is identified by its name. The initially created signals are incomplete.

Select an output at a grouping entry and press the right mouse button. Execute the menu command **connect signal**. A pop–up menu appears providing a list of signal names you can connect the output to. If you choose one of them, the output is renamed. The same works if you select an input of a logical grouping. Note that the provided list of signals is filtered with respect to type compatibility.

To rename a signal, select the i/o–item and execute **rename signal**. Type in the new name. Note that the name is accepted only if it does not already identifies another signal in the logical grouping. The entire signal is renamed (i.e. all i/o–items of grouping entries representing the signal are renamed).

The command **break signal** allows to disconnect an input or output of a grouping entry from a signal. In fact, you have to create a new incomplete signal that is represented exclusively by the selected i/o–item. A text field appears inviting you to specify the new signal's name. The name is accepted only if it does not already identifies another signal in the logical grouping.

To connect an atomic output to atomic inputs via a logical grouping, you have to connect the atomic output to a signal (i.e. an input to a logical grouping). Then you have to connect the atomic inputs to the same signal (i.e. outputs from the logical grouping that have the same name than the signal's input).

You can get a list of all signals contained in a logical grouping by selecting a grouping entry or a grouping link associated to a composite function block and executing the menu command **show signals**.

Copyright per DIN 34

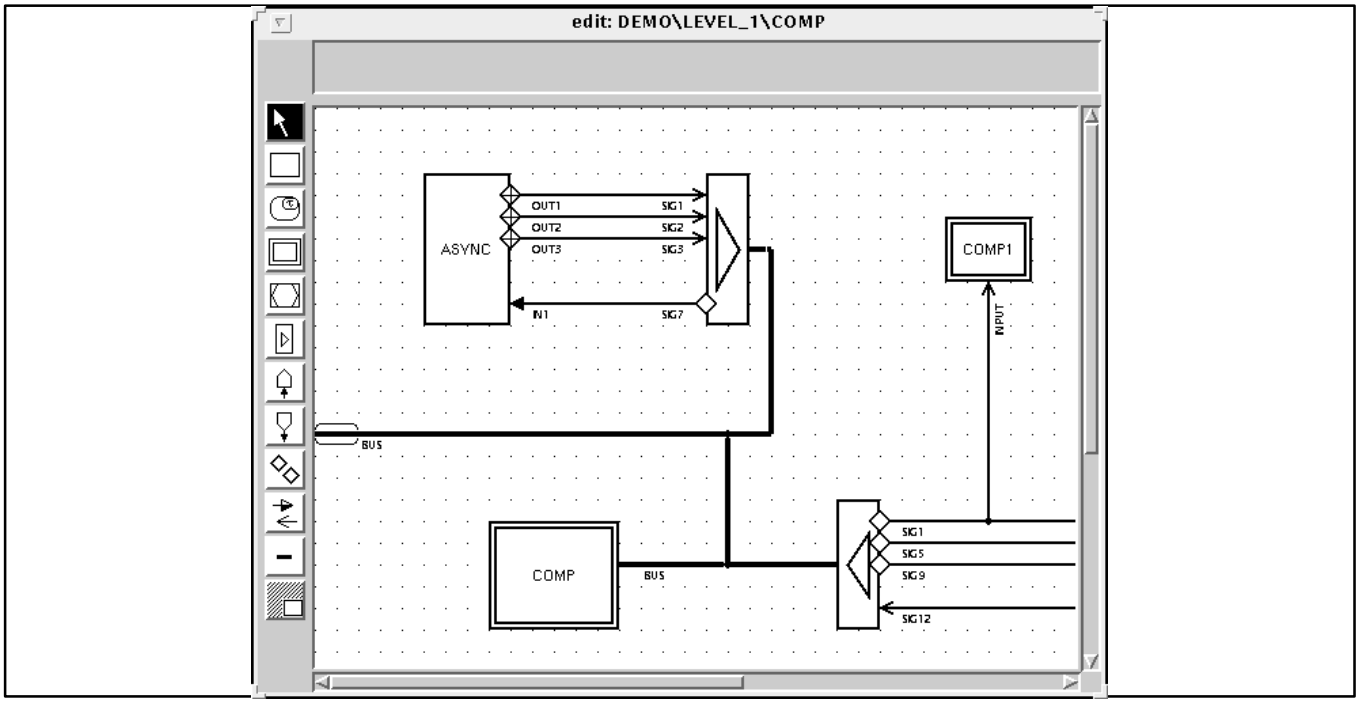


Figure 6–51 : *A Composite Editor showing an example of a logical grouping.*
Note that ASYNC\OUT1 is connected to COMP\INPUT via logical grouping (signal SIG1)

If you connect two logical groupings, the system checks the signals of both groupings. It's not allowed to have two signal inputs with the same name in a logical grouping. The signals are also checked on type compatibility. If errors are found, an error window is opened providing error descriptions; the operation of connecting the two logical groupings becomes aborted.

6.4.7.1.31 Global Symbols

Global symbols provide another convenient way of connecting data sources (top level inputs, atomic outputs) to data sinks (top level outputs, atomic inputs). A global symbol consists of one definition point and a number of reference points that are identified by the same name (the global symbol's name).

*The definition point of a global symbol has to be defined either **inside the same** or on a **higher level** composite function block than the reference points of this global symbol.*
This is called the visibility rule for global symbols.

Create a number of definition points and reference points. Select one of the reference points and press the right mouse button. Execute the menu command **connect**. You get a list of the names of all definition points that you can connect the reference point with. Choose one of them and see that the reference point is renamed to the selected name.

Create a composite function block and step into it. Create further reference points and connect them to the previously defined definition points.

Copyright per DIN 34

To disconnect a reference point from a definition point, select the reference point and execute the menu command **break reference**. Automatically a new name is generated for the selected reference point that does not reference a visible definition point (i.e. a definition point defined in the same or a higher level composite function block).

You can rename an entire global symbol by selecting its definition point and executing the menu command **rename**. A text field appears allowing you to type in the new name. If the name is ok (does not already identify another object), the definition point and all connected reference points are renamed.

If a definition point is selected, the menu you obtain by pressing the right mouse button provides a command **show references**. On executing this command, you get a list of the pathnames of all connected reference points (including their type).

To connect an atomic output to atomic inputs via global symbol, you have to connect the atomic output to the definition point. Then you have to connect the atomic inputs to the global symbols' reference points.

The list of visible definition points you get when you want to connect a reference point to a definition point provides only those definition points that are type compatible. However, it may be possible that the types of data source and data sinks connected to a global symbol differ. For example, if you cut a definition point, its associated reference points may become associated to a definition point with the same name on a higher composite hierarchy level (if there is one); this may lead to type incompatibilities in the global symbol.

If a definition point or a reference point is selected, the pop–up menu obtained by pressing the left mouse button provides a command **unify type**. On executing **unify type**, a list of all types occurring in the global symbol appears. If you choose one of them, the type of all connected data items is set to the selected one.

Copyright per DIN 34

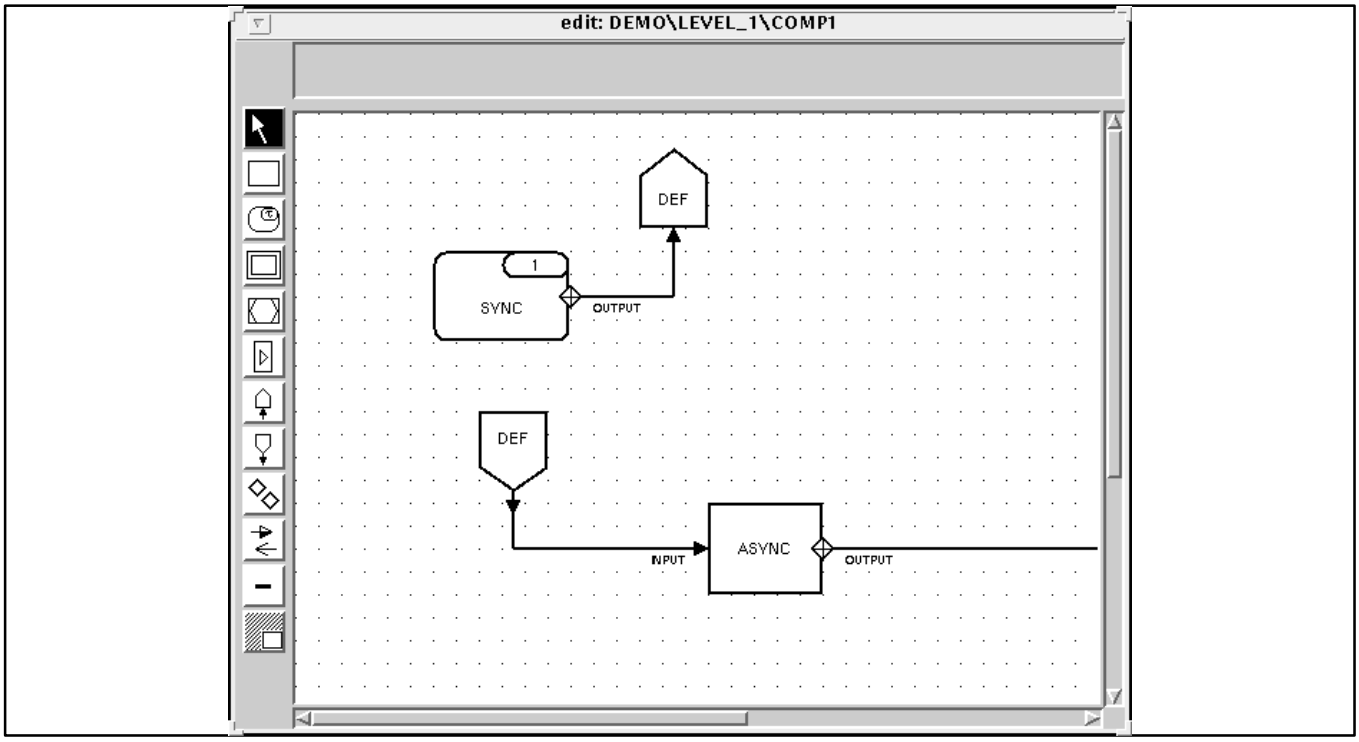


Figure 6–52 : *A Composite Editor showing an example of a global symbol. Note that SYNC\OUTPUT is connected to ASYNC\INPUT via global symbol DEF.*

6.4.7.1.32 Editing an Atomic Function Block

Atomic function blocks can be implemented by AIL–code (AIL–code is defined as a subset of the Ada language) or by decision table.

Select an asynchronous or a synchronous function block. Execute the menu command **edit**. A submenu appears allowing you to open either an AIL–Editor or a Decision Table Editor (submenu items **AIL code** resp. **decision table**).

If you try to open an AIL–Editor (resp. Decision Table Editor) on an atomic function block that is implemented by decision table (resp. AIL code), a confirmer appears warning you that the current implementation may become lost.

The use of following Ada features is **not allowed** within AIL code:
access types, package declaration, Ada tasking, address clauses

- *For more information about data types refer to section 6.4.7.1.37.2 and chapter 6.4.12.*
- *For more information about mathematical constants and routines refer to chapter 6.4.11.*

6.4.7.1.33 Changing the Grid

All operations including the determination of objects' positions (moving, pasting) are restricted to the current grid. The default grid is indicated by the dot pattern of the graphic area.

Copyright per DIN 34

When there are lots of connections and you want to move some connection lines you may find out that the default resolution is not fine enough. So it may be impossible to find a suitable place that is not occupied by other connection lines. Choose the basic menu command **grid align**. A submenu appears listing the available grid values: **default**, **1/2** (* default), **1/4** (* default). Choose a finer grid and try some moving operations.

- *Note that the dot pattern indicates the default grid, not the current grid.*
- *Note that i/o–items minimal spacing is restricted to default grid to avoid overlapping.*

6.4.7.1.34 Changing the Size of a Block's Inside View

The default size of the inside view of a composite function block is A5. While editing you may find out that you want to enlarge the size.

Choose the basic menu item **size**. A submenu appears listing the available sizes: **A5, A4**, **A3** and **A2**. Choose the desired size and continue editing.

- *Note that a reduction of size affords that a part of the document has to be cut off. This part has to be empty. Otherwise the operation will be aborted indicated by a message displayed in the message subview*

6.4.7.1.35 Searching for an Object

Choose the basic menu item **find**. A submenu appears asking you if you want to search for a function block, constant block, global symbol or for an i/o–item. Execute the submenu command **function** or **i/o**.

If you execute **function**, a pop–up menu appears listing all function blocks, constant blocks, global symbols defined on the actual composite level. The objects are identified by their names. Choose the object you want to find. See that the object becomes selected. If it is not already visible inside the graphic subview, the subview will be scrolled automatically.

The submenu command **i/o** works in a similar way. The appearing pop–up menu lists the names of all i/o–items defined at the currently edited composite function block (i.e. i/o–items to the next higher level seen from inside).

6.4.7.1.36 Changing a Function Block's Type

Select an asynchronous function block and press the right mouse button. Choose **change type**. A submenu appears listing the possible function block types, in this case **synchronous** and **composite**.

Similarly a synchronous function block can be converted to an asynchronous or a composite function block; a composite function block can be converted to an asynchronous or a synchronous function block.

- *Note that you may loose some of the information belonging to the selected object. If you convert an atomic function block to a composite function block, the implementation gets lost as well as the activa-*

Copyright per DIN 34

tion characteristics of the associated inputs, for example. On the other hand, if you convert a composite function block to an atomic function block, the contents of the composite can't be kept.

6.4.7.1.37 Defining Variables

If an output or an input is selected, the pop–up menu provides an item **variable**. If the cursor is positioned over **variable**, a submenu appears listing the details that can be specified to define the variable.

Each variable is to be defined by a type . On executing the submenu command **type**, a pop–up menu appears providing a list of possible types. The contents of the list of possible types depends on wether you selected a Top Level I/O (Inputs/Outputs connected to the border of the Top Level function block) or I/Os connected to function blocks within the model.

Variable Type	***Top Level I/O***	***Function Block I/O***
Unsigned_Byte	*X*	*X*
Signed_Byte	*X*	*X*
Unsigned_Short_Word	*X*	*X*
Signed_Short_Word	*X*	*X*
Unsigned_Integer	*X*	*X*
Signed_Integer	*X*	*X*
Real	*X*	*X*
Long_Real	*X*	*X*
Boolean	*X*	*X*
State_Code	*X*	*X*
Pulse	*X*	*X*
Burst_Pulse	*X*	*X*
Complex	*not available*	*X*
Time	*not available*	*X*
Long_Duration	*not available*	*X*
Vector	*not available*	*X*
Matrix	*not available*	*X*
Record	*not available*	*X*

Figure 6–53 : *A list of all available data types*

Figure 6–53 provides a list of the available types. The latter six types are available only within the model. Onboard types (top level inputs, top level outputs) have to be one of twelve types listed first.

Depending on the selected i/o–item (atomic input/output, composite input/output, model input/output) and the i/o–item's type the submenu that appears on executing **variable** provides the appropriate commands to define the variable.

Copyright per DIN 34

If the variable type is VECTOR, MATRIX or RECORD separate input windows appear which allow to define the selected data type.

If the type is defined, you can specify the initial value (submenu command **initial value**). Depending on the variable's type, a text field or a pop–up menu appears allowing you to designate the initial value. The value is checked on correctness, i.e. the initial value setting includes a type check.

The write access can be specified by executing the submenu commands **enable write access** or **disable write access** (set resp. reset the write access marker).

- *Note that all outputs are automatically created with the write access marker set.*

If the variable type is REAL or LONG_REAL, the menu provides a field **engineering unit**. A text field allows you to specify the engineering unit.

- *Connections can only be established between type compatible i/o–items. In general, type compatibility is given if the type identifier is the same.*

6.4.7.1.37.1 STATECODE,VECTOR,MATRIX,RECORD as I/O

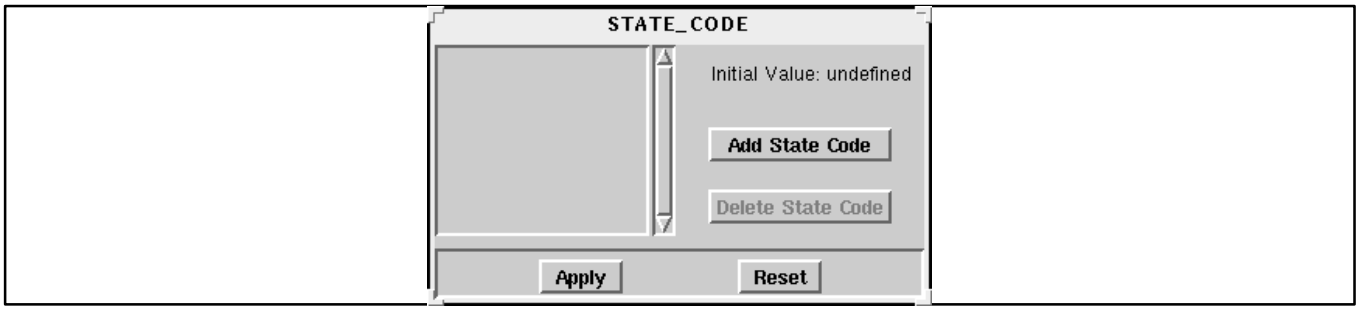


Figure 6–54 : *The empty state code definition window*

Creating a state code variable

- Select the desired input/output by clicking on it with the left mouse button.
- Press and hold the right mouse button. The related pop–up menu appears.
- Move the mouse pointer to the **variable–>type** command and release the mouse button. A list of all available data types appears.
- Move the mouse pointer to the entry **STATE_CODE** and press the left mouse button.
- The definition window pops up (see Figure 6–54). Press the **Add State Code** button.
- Type the new state code in the input line. (see Figure 6–55) and press the **Accept** button. The state code is listed in the list subview in Figure 6–54.
- *Note that the state code is locked up in double quotes , the string has to be filled with blanks to fit into the 8 characters limit.*
- Repeat the preceding steps to enter more state code definitions.
- Press the **Apply** button in the state code definition window.

Copyright per DIN 34

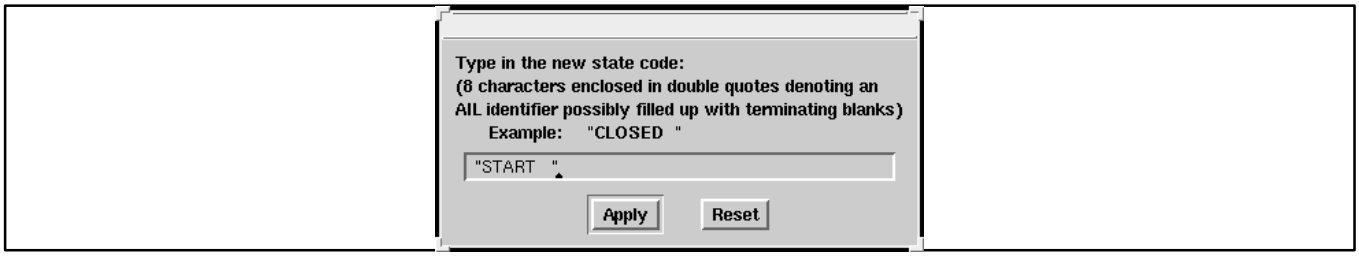

Type in the new state code:
(8 characters enclosed in double quotes denoting an
AIL identifier possibly filled up with terminating blanks)
Example: "CLOSED "
"START "
Apply
Reset

Figure 6–55 : *The state code input window*

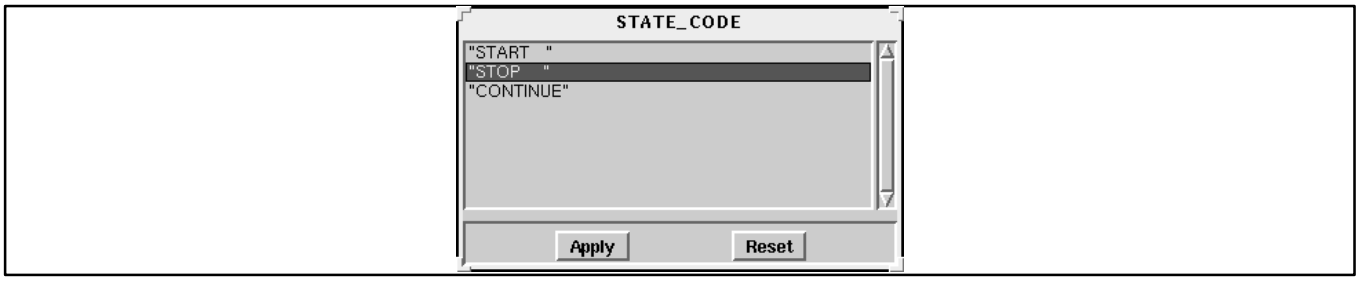


Figure 6–56 : *The initial value selection window for state code variables*

Setting the initial value for state code parameters

- Select the desired input/output by clicking on it with the left mouse button.
- Press and hold the right mouse button. The related pop–up menu appears.
- Move the mouse pointer to the **variable–>initial value...** command and release the mouse button. The initial value selection window (see Figure 6–56) appears.
- Select the desired initial value by clicking on the name in the list. The selected value becomes highlighted.
- Press the **Apply** button.

For the data types vector, matrix and record some additional rules appear.

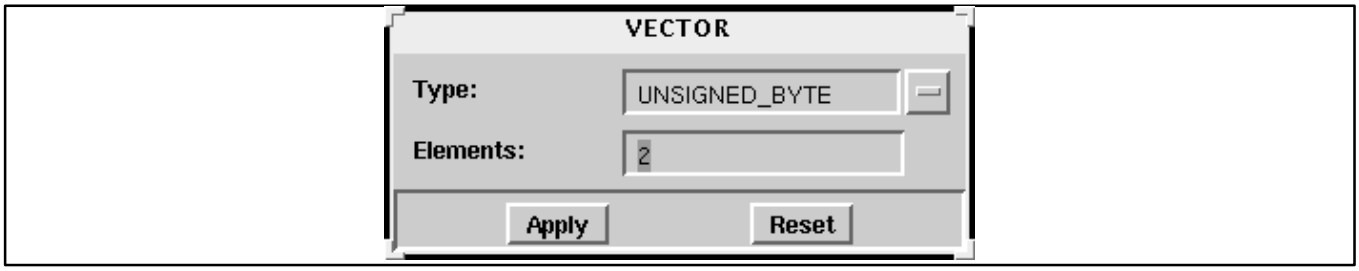


Figure 6–57 : *Definition window for a variable of type vector*

After definition of a variable of type vector the window as shown in Figure 6–57 appears. The definition window for vector types shows the default number of elements for a vector (which is at least 2). Type the number of vector elements you need, then press the **Type** button. A predefined list of available data types for the vector elements pops–up as shown in Figure 6–58. Select one of the data types, then press the **Apply** button. In the example above a vector variable with 2 elements, both of type REAL is created.

Copyright per DIN 34

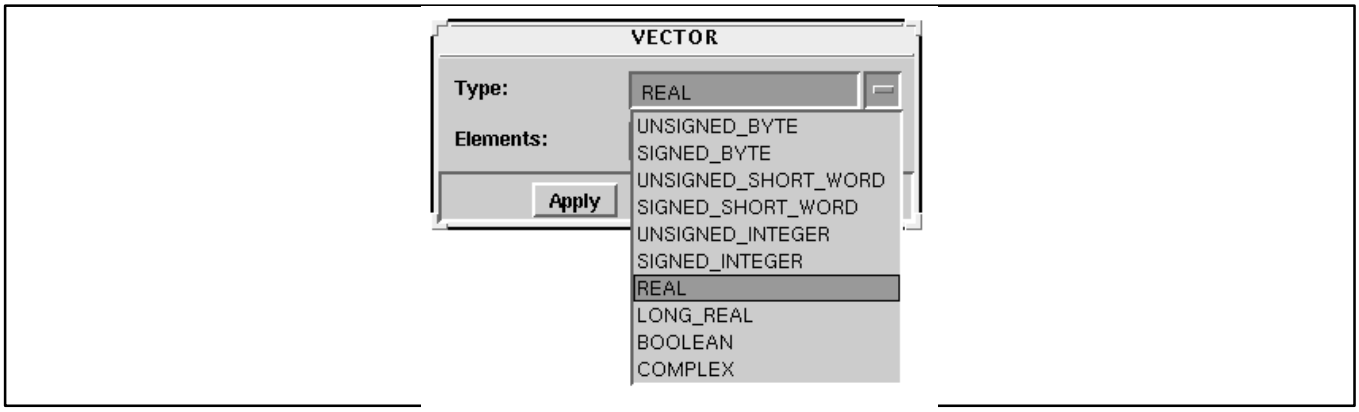


Figure 6–58 : *Definition windo efined list of data types.*

- *Note that you can choose only one data type for all elements of a vector.*

After definition of a variable of type matrix the matrix definition window appears. Change the number of rows and columns accordingly. The press the **Type** button. Select the data type for the matrix elements from the predefined list then press the **Apply** button.

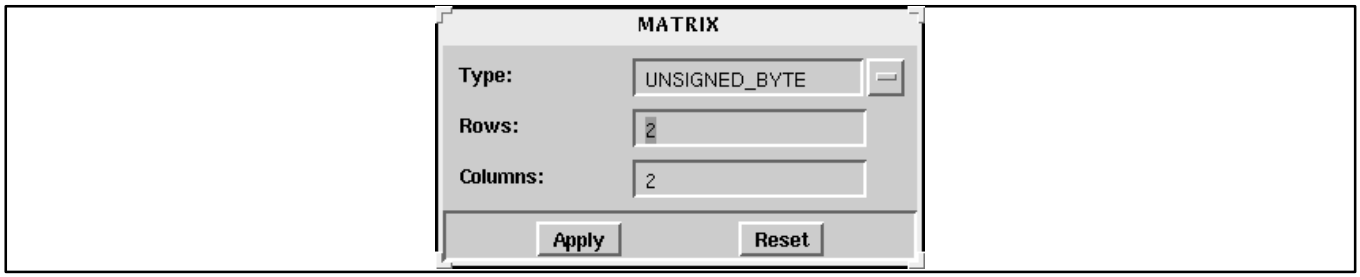


Figure 6–59 : *Definition window for a variable of type matrix with predefined rows and columns.*

- *Note that you can choose only one data type for all elements of a matrix.*

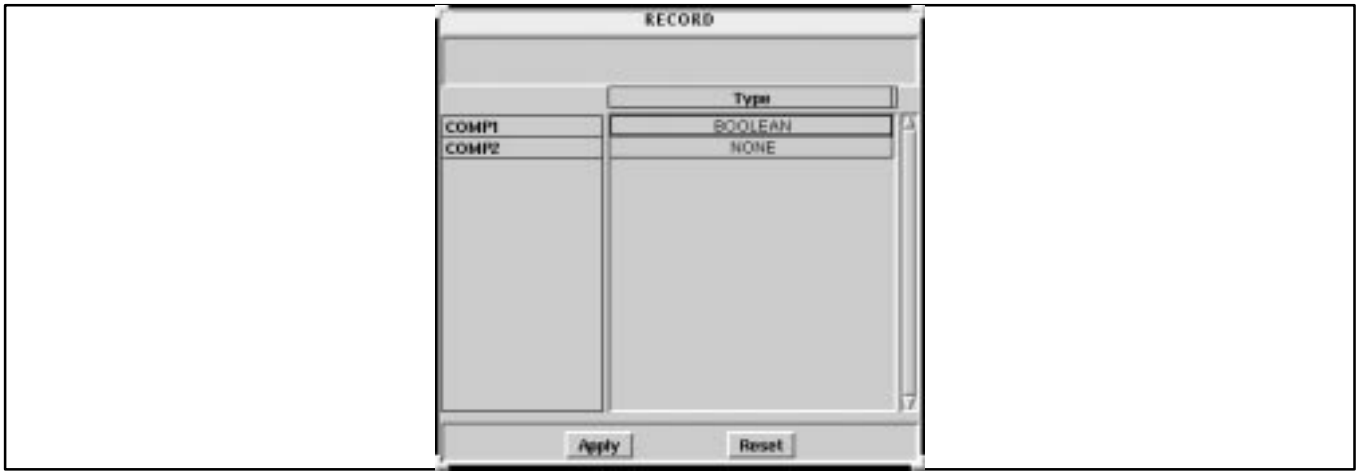


Figure 6–60 : *Definition window for a variable of type record.*

- *Note that the data type record is only a graphical combination of different variables.*

Copyright per DIN 34

Figure Figure 6–60 shows the definition window for the record data type. Select the field COMP1. This is the default name for the first component. The pop–up menu command **rename** allows to set a new name for this component. With the command **insert component –> preceding** or **insert component –> succeeding** you can add further components before or behind the selected one. The pop–up command **delete** removes the selected component.

With no component selected the pop–up menu provides the commands **undo** and **add component**.

To change the type of a component select the field in the column Type. Select the data type from the list of predefined types.

- *Note that each component in the record can be of different data type.*
- *For more information about data types refer to section 6.4.7.1.37.2.*

6.4.7.1.37.2 The AIL data types

This section describes the AIL data types used for the implementation of AIL code and the connection between the function blocks.

Following informations will be listed:
Data Type, the internal representation, the range, further informations i.e. examples if necessary

- *For the detailed Ada specification of all available AIL data types and their available functions refer to chapter 6.4.12.*

UNSIGNED_BYTE

internal representation: 8 bit unsigned integer
range: 0 to +255

SIGNED_BYTE

internal representation: 8 bit unsigned integer
range: –128 to +127

UNSIGNED_SHORT_WORD

internal representation: 16 bit unsigned integer
range: 0 to +65535

Copyright per DIN 34

SIGNED_SHORT_WORD

internal representation: 16 bit unsigned integer
range: –32768 to +32767

UNSIGNED_INTEGER

internal representation: 32 bit unsigned integer
range: 0 to +4294967295

Note that in the current implementation the upper limit is +214748367

SIGNED_INTEGER

internal representation: 32 bit unsigned integer
range: –2147483648 to +2147483647

REAL

internal representation: 32 bit IEEE single float
range:

LONG_REAL

internal representation: 64 bit IEEE single float
range:

BOOLEAN

internal representation: enumeration type
default values: TRUE, FALSE

STATE_CODE

internal representation: 8 character string

State code naming conventions are:

– the first character must be a letter
– any subsequent characters must be letters, digits, or the underscore ('_')

Copyright per DIN 34

– two underscores cannot occur together, nor can a state code end with an underscore

– all state codes will be in capital letters only

PULSE

internal representation: **BOOLEAN**

REMARK:
A pulse can be set to TRUE during model run–time via MOCS, it will be automatically reset after the execution of the time frame the pulse is associated with.
Only one pulse will be generated.

BURST_PULSE

internal representation: **UNSIGNED_INTEGER**

REMARK:
The burst pulse number can be set during model run–time via MOCS, it will be automatically reset after the execution of the time frame the burst pulse is associated with.

COMPLEX

internal representation: re: **LONG_REAL**, im: **LONG_REAL**

Note that you can choose between GAUSS or POLAR notation in the input window display.

TIME

internal representation:
year: INTEGER, range: 1901 .. 2099,
month: INTEGER, range: 1 .. 12,
day: INTEGER, range: 1 .. 31
seconds past midnight: DURATION

Note that the user enters the seconds past midnight in a simplified input format:
hour: INTEGER, range 0 .. 23,
minute: INTEGER, range 0 .. 59,
second: float, range 0.00 .. 59.99

EXAMPLE: 1.1.1993 12:17:59.12

LONG_DURATION

Copyright per DIN 34

internal representation: DURATION
range: –86400.00 sec to 86400.00 sec

Note that in the current implementation the internal representation is fixed point

VECTOR

internal representation: 1–dimensional array of a scalar type
range: max 255 elements

For the vector elements following scalar data types are allowed:

UNSIGNED_BYTE,
SIGNED_BYTE,
UNSIGNED_SHORT_WORD,
SIGNED_SHORT_WORD,
UNSIGNED_INTEGER,
SIGNED_INTEGER,
REAL,
LONG_REAL,
BOOLEAN,
COMPLEX

MATRIX

internal representation: 2–dimensional array of a scalar type
range: max 255 elements **in total**

For the matrix elements following data types are allowed:

see above VECTOR

RECORD

is a **graphical** association of inputs/outputs.

internal representation: is the internal representation of the record elements
range: The number of elements in a record should be less than 200.
The accurate number can not be given because it depends on the total number of
inputs/outputs implemented.

For the record elements following data types are allowed:

Copyright per DIN 34

see above VECTOR

- *In AIL code you have access to the element of a record via record name – underscore – element name.*
  EXAMPLE:
  The name of the record is CABLE with the elements POWER, STATUS and VOLTAGE.
  You can access the three values via CABLE_POWER, CABLE_STATUS and CABLE_VOLTAGE.

6.4.7.1.38 Performing a Rule Check

The basic menu command **check** allows to check a simulation model on correctness. A window appears inviting to specify the checks to be performed by selecting the corresponding buttons.

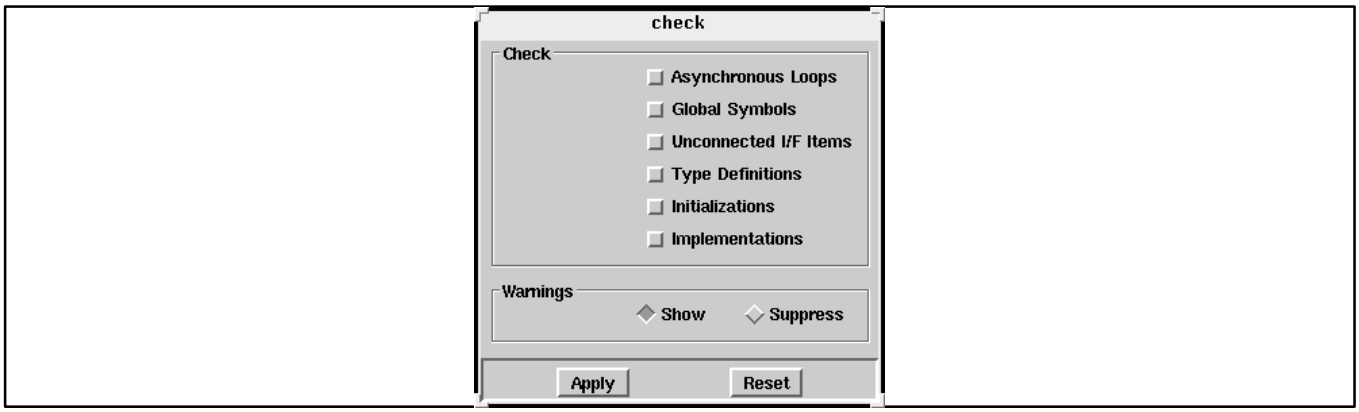


Figure 6–61 : *The Rule Check Parameters window*

To check the model on the existence of asynchronous loops, select the button **Check: Asynchronous Loops**. The completeness and correctness of the model's global symbols can be verified by selecting **Check: Global Symbols**. To check whether all atomic and composite function blocks are implemented, select **Check: Implementations**.

The model's variables can be checked on type definitions (select **Check: Type Definitions**) and on initializations (select **Check: Initializations**). Unconnected interface items are found by selecting **Check: Unconnected I/F Items**.

On pressing the **Apply** button the checks are performed, possibly resulting in the generation of warning and/or error messages. It is possible to suppress the display of warnings by selecting the button **Warnings: Suppress**. Otherwise, if warnings should be displayed, select **Warnings: Show**. If errors (and/or warnings) occur, an error window is opened providing detailed information.

6.4.7.1.39 Saving the Model

To save the current state of your work, execute the basic menu command **save**. The cursor changes shape to that of a pen indicating that all the modifications are written.

Copyright per DIN 34

6.4.7.1.40 Model Compilation and Simulator Kernel Configuration

6.4.7.1.40.1 Compiling atomic function blocks

The basic menu command **compile** allows to compile one or more atomic function blocks separately. You can select the atomic functions blocks to be compiled on the current level. If you select a composite function block all function blocks on lower levels in this composite are compiled. If there is no function block selected, all atomic function blocks in the model are compiled.

A window appears allowing to specify the compilation parameters. The **Architecture** button allows to select the target architecture. The **Host** button allows to select the target host on which the compilation shall be performed. The architectures and host names provided correspond to the specifications in the system's CSS configuration file.

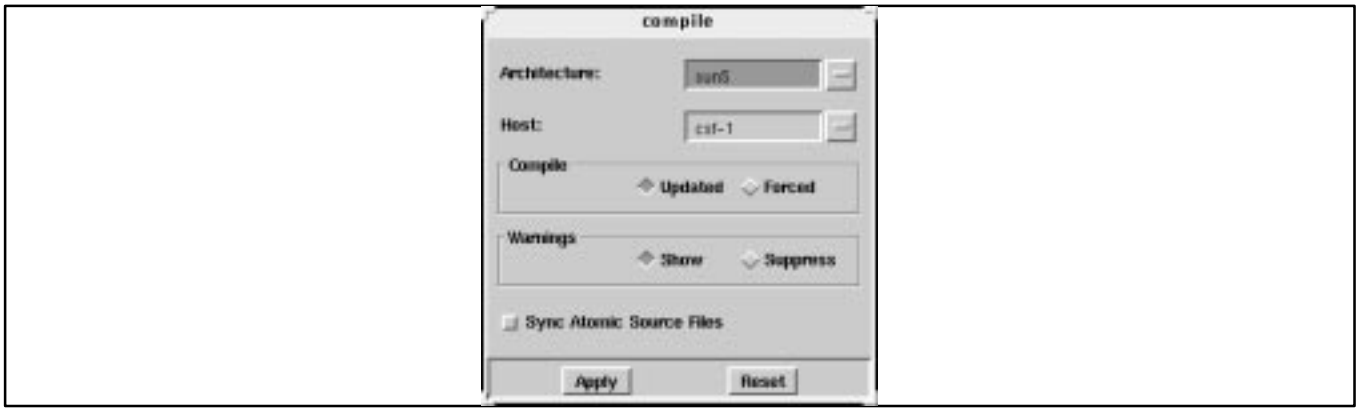


Figure 6–62 : *The Compilation Parameters window*

To reduce the time consuming compilation phase, incremental compilation is provided. Incremental compilation means only those atomic function blocks which have been edited since the last compilation will be compiled. Select the button **Compile: Updated** to perform incremental compilation. By selecting **Compile: Forced** all selected atomic functions blocks will be compiled.

In rare conditions, the generated Ada source files corresponding to the atomic function blocks may be corrupt. If the button **Sync Atomic Source Files** is selected, the system writes, prior to compilation, the source code of all atomic function blocks once again to the file system. However, this operation, that may be time consuming for bigger models, is necessary only in exceptional conditions (e.g. if someone has manipulated the source files in the file system).

When both **Compile: Updated** and **Sync Atomic Source Files** is selected, the system will also first purge the complete architecture–specific file system branch for the model, including executable and Ada library, which does no additional harm in this case and may occasionally be necessary to recover from a crashed compilation.

Copyright per DIN 34

Prior to compilation, an MDE–GL rule check, possibly resulting in the generation of warning and/or error messages, will be performed. It is possible to suppress the display of warnings by selecting the button **Warnings: Suppress**. Otherwise, if warnings should be displayed, select **Warnings: Show**.

On pressing the **Apply** button, a confirmation window may appear advising to save the model if there are any pending modifications. If saving is not confirmed, the operation is aborted. Then the rule check is performed, and, if any errors are found, the operation is aborted. An information window is opened showing the errors (and/or warnings) generated by the rule check, if there are any. Another information window provides the compilation status and lists the warnings and/or errors which occur during compilation.

The compilation information is additionally stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension .COMPILA TION_REPORT_TXT . The part of the name preceeding the extension matches the same pattern as for log and archive files (refer to section 6.4.7.9.4 for more information on names of log and archive files).

6.4.7.1.40.2 Simulator Kernel Configuration

The basic menu command **configure–>simulator kernel** compiles the model's atomic function blocks, configures an executable simulator kernel (i.e. generates and compiles a runtime system and links the atomic function blocks with the runtime system) and generates the initial simulation state.

A window appears allowing to specify the compilation parameters (target architecture, target host, enable/disable incremental compilation, suppress/show warnings, sync atomic source files) as described in the previous section.

On pressing the **Apply** button, a confirmation window may appear advising to save the model if there are any pending modifications. If saving is not confirmed, the operation is aborted. Then a complete rule check is performed. An information window is opened showing errors (and/or warnings), if there are any. If there are any errors found, the operation is aborted.

While simulator kernel configuration is performed, the system provides progress information in an information window. You are informed whether the simulator kernel could be configured successfully or errors have led to breaking off the operation. Note that the action of simulator kernel configuration may be time consuming.

The configuration progress information is additionally stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension .COMPILA TION_REPORT_TXT . The part of the name preceeding the extension matches the same pattern as for log and archive files (refer to section 6.4.7.9.4 for more information on names of log and archive files).

If the simulator kernel configuration terminated successfully, you can start a simulation using MOCS.

Copyright per DIN 34

Note that an initial simulation state is created during simulator kernel configuration. This simulation state sets the simulation to the start conditions.

The simulation state defines the start–up state of your simulation. If you are not interested in starting the simulation just from the initial state you can use MOCS to create more simulation states.

6.4.7.1.40.3 Adaptation System Configuration

If the model is intended to be used for H/W in the loop simulations, i.e. it's interface references onboard items in the database, the basic menu command **configure–>adaptation system** allows to generate new configuration files (one for each architecture a simulator kernel is configured for) for the Command and Measurement Adaptation System (CMAS). All onboard items referenced by the model are checked on completeness and consistency. If warnings or errors occur, an information window providing detailed information is opened.

Since definitions of onboard data and adaptation requirements (i.e. external interfaces of the simulator) are usually project specific, configuration for a project specific CMAS is supported by checking for the existence of an executable file **$CMAS_HOME/bin/common/start_db_server.** If present, this executable will be started instead of the internal standard configuration routine and given the responsiblity to generate a proper CMAS configuration file.

In this case, the rules for correct definition of referenced onboard end items as specified in section 6.4.10 will obviously be overruled by project specific definitions.

Each time the definitions of referenced onboard items have been modified in the database or the simulation model has been edited and a new simulator kernel has been configured, new configuration files have to be generated.

6.4.7.1.41 Printing Out a Document

Make sure that your laser printer is activated. Execute the basic menu command **print**. The documentation of the currently edited composite function block is printed out (graphic and i/o–list). The previous selection of more than one function block leads to a related documentation printout.
Note that the process of printing takes a little time.

For detailed information about documentation functions see chapter 6.4.7.7 (Documentation Generation Function).

6.4.7.2 Model Inspectors

For each function performed with a dedicated editor tool there is also an inspector tool provided. The inspector tools provide all functions specific to the editor except the function to make changes. For example see Figure 6–63 which shows the composite inspector window. Note that the tool button column on the left side is missing.

Copyright per DIN 34

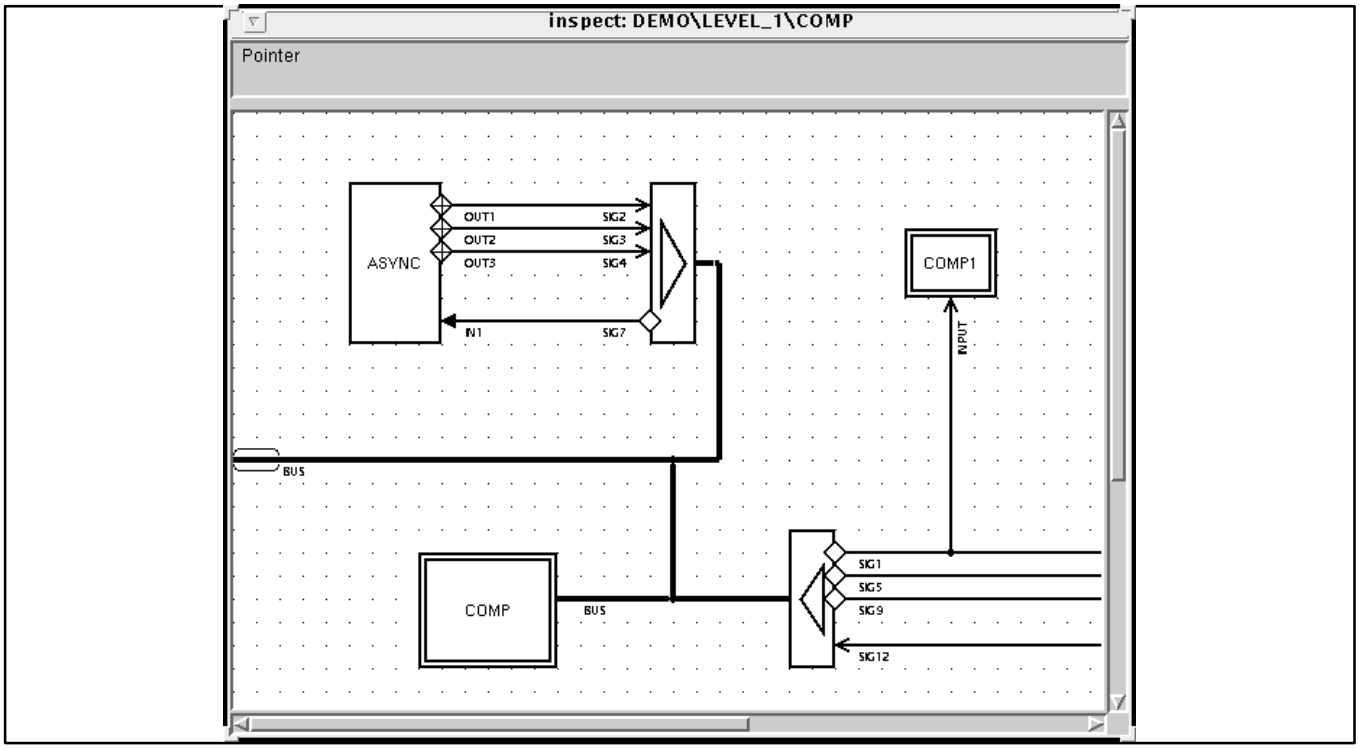


Figure 6–63 : *Composite Inspector User Interface*

The inspector tools are used to inspect items and to create documentation without changing the items.

Copyright per DIN 34

6.4.7.3 Composite Interface Editor

The Composite Interface Editor runs in a separate MDE window. It may be activated from within the Data Base Browser or a Composite Editor. The tool allows to inspect and edit the interfaces of composite function blocks.

The Composite Interface Editor provides a graphic subview showing the composite function block including its interface and scrolling list subviews for all kinds of interface items (inputs, outputs, grouping links). A further scrolling list subview is used to present all signals of a selected logical grouping (if no grouping link is selected, this subview stays empty).

The interface items can be selected interactively either inside the graphic subview (selecting works in a similar way as in the Composite Editor) or in one of the scrolling list subviews. All these subviews provide selection sensitive pop–up menus to edit the interface.

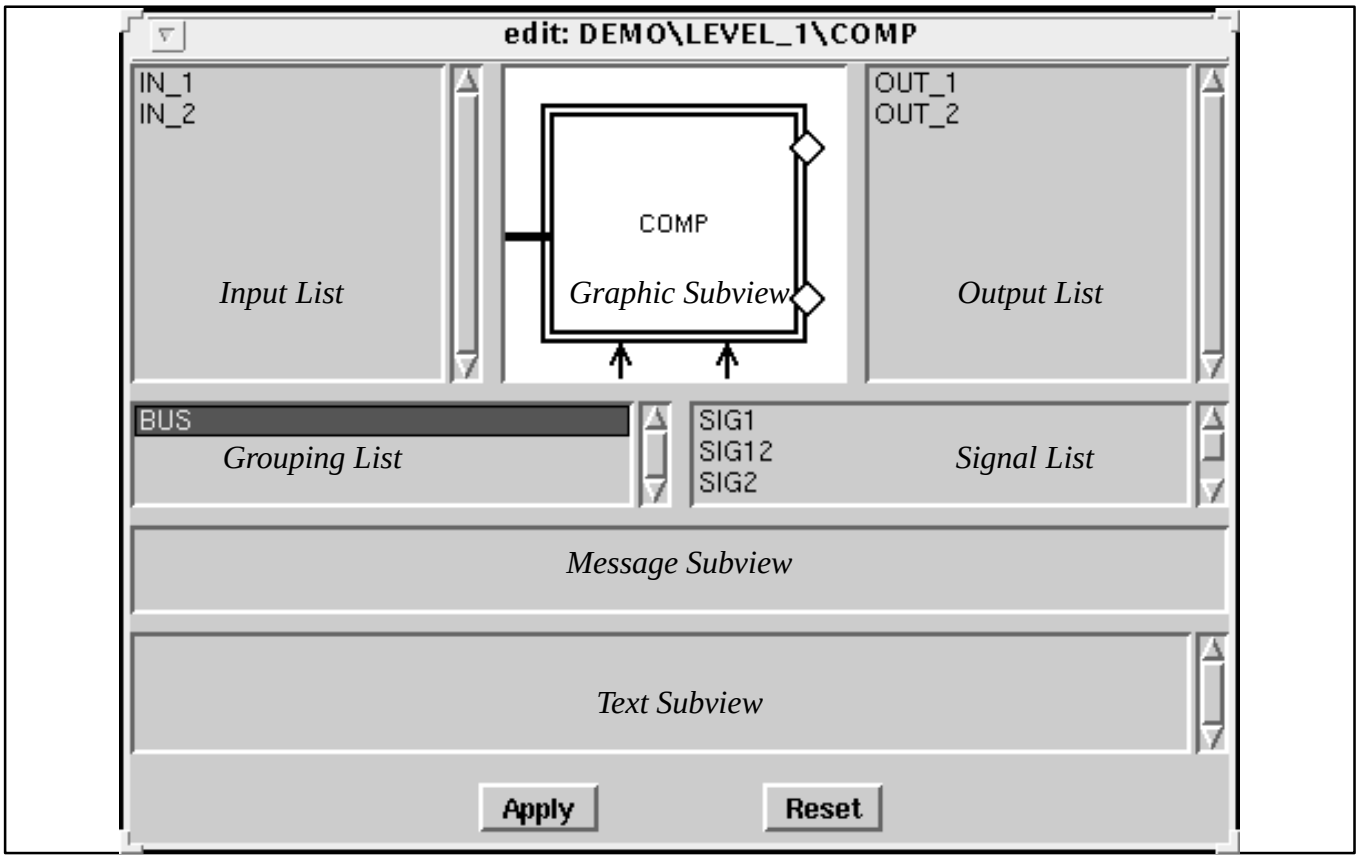


Figure 6–64 : *Composite Interface Editor user interface*

The main edit functions provided in the graphic subview and the scrolling list subviews are listed briefly:

* creating interface items (only inside the graphic subview)
* moving interface items (only inside the graphic subview)
* cutting interface items
* copying/pasting interface items (pasting only inside the graphic subview)
* copying the composite function block (to a Composite Editor)
* changing the attributes of interface items (e.g. type, initial value, etc.)

Copyright per DIN 34

* generating and printing documentation

The Composite Interface Editor provides a text subview that can be run in inspect mode (read only mode). Usually if an interface item is selected, the text subview provides information about the selected item in read only mode.

If there is no interface item selected, the text subview shows the comment of the composite function block. In this case the view runs in edit mode allowing to change the comment. It provides the common text editing functions. These comprise:

* typing in text
* selecting text
* copying text (possibly cross text windows)
* cutting text

It is possible to copy the edited block to a Composite Editor. Furthermore the Composite Interface Editor allows to copy/paste interface items from and to a Composite Editor, another Composite Interface Editor, an Atomic AIL Editor or an Atomic Decision Table Editor.

The user may generate and print out documentation of the composite function block. The following outputs are provided:

* textual output describing a selected i/o-item (e.g. type, initial value)
* textual output of the composite function block's comment

User Interface

Figure 6–64 shows the user interface of the Composite Interface Editor. It provides scrolling list subviews (*input list, output list, grouping list and signal list*) and a graphical subview allowing interface item selection (*graphic subview*). Note that the signal list will be displayed after selection of an item in the grouping list. A read only text subview is used to provide textual feedback to the user (*message subview*). A text subview that can be run in read only mode as well as in write mode is used to give textual information of the selected interface item (*text subview*). When there is no interface item selected, the *text subview* runs in edit mode allowing to inspect/edit the comment of the composite function block.

Editing of the interface is done inside the *input list, output list, grouping list* and *graphic subview* via selection sensitive pop–up menus.

Saving the edited text within the text subview (comment) is done clicking on the **Apply** button, whereas via the **Reset** button one goes back to the last saved version.

Copyright per DIN 34

6.4.7.4 Atomic Editors

The Atomic Editors are the tools to edit atomic function blocks. Since atomic function blocks can be implemented in two different ways (either by AIL code or by decision table), there are two kinds of Atomic Editors: the AIL–Editor and the Decision Table Editor.

You can open an Atomic Editor to edit an atomic function block from inside the Composite Editor. This chapter explains the different features of this tool.

6.4.7.4.1 Common

Using the Composite Editor, create several atomic function blocks. Assign several i/o–items to them. Establish connections between the i/o–items. Then select one of the atomic function blocks and execute **edit** and the submenu command **AIL code** to open an AIL Editor. Similarly, open a decision table editor on another atomic function block (submenu command **decision table**).

6.4.7.4.2 Components

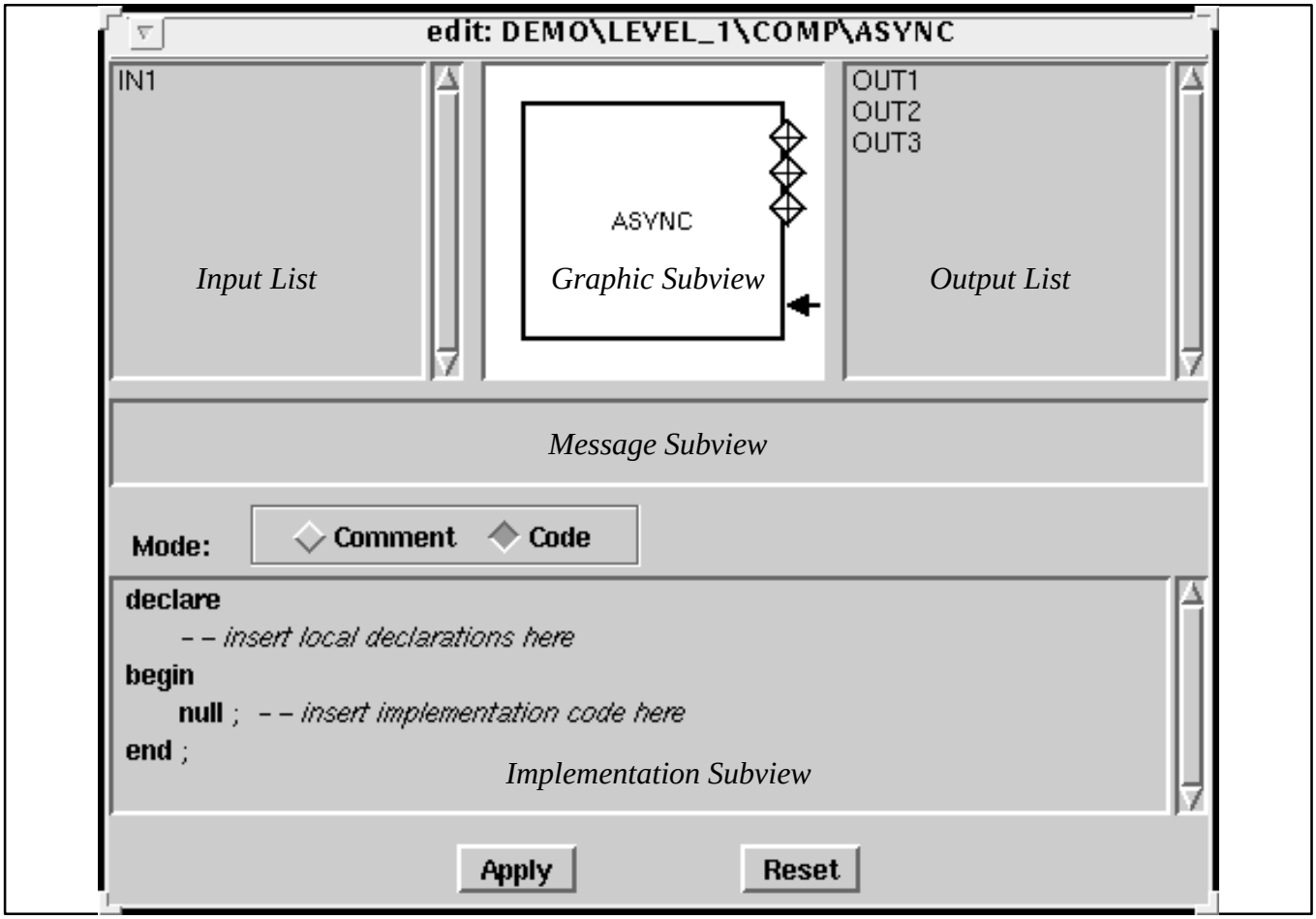


Figure 6–65 : *Components of the Atomic Editor*

Copyright per DIN 34

6.4.7.4.2.1 The Label

The label of the atomic editor shows the pathname of the atomic function block.

6.4.7.4.2.2 The Graphic Subview

The graphic subview shows the graphical symbol of the atomic function block including all of its i/o – items. Note that connections are not displayed.

You can select the i/o – items quite similar like you used to do using the Composite Editor. So move the cursor over an output and click the left button. Note that accordingly the text subwindow changes its contents. It shows information about the selected i/o–item.

Note also that simultaneously in the output list the name of the selected formal out – parameter is highlighted. It is not possible to select more than one i/o–item at a time. To deselect move the cursor to a location so that it doesn't point to an i/o–item and click the left button.

You can edit the selected i/o–item by pop–up menu commands the same way as with the Composite Editor.

6.4.7.4.2.3 The Input and Output Lists

The input list shows the names of all inputs; the output list shows the names of all outputs. Because the subwindows may not be large enough do show the whole lists, a scrollbar is associated.

You can select an input or output by moving the cursor over its name and clicking the left button. Note that simultaneously in the graphic subview the appropriate i/o–item is selected. Information about the selected object is shown in the text subview.

To deselect, move the cursor over the highlighted name and click the left button. The i/o–item becomes deselected.

You can edit the selected i/o–item by pop–up menu commands the same way as with the Composite Editor.

6.4.7.4.2.4 The Message Subview

While editing the message subview is used to provide feedback. For example if you try to perform an illegal action you will be informed by a message indicating the mistake.

6.4.7.4.2.5 The Implementation Subview

Since atomic function blocks may be implemented either by AIL code or by decision table, the implementation subview of the AIL–editor is different from the implementation subview of the decision table editor.

Copyright per DIN 34

However, in both editors, if there is an i/o – item selected, the implementation subview is used to provide information about this particular i/o – item. In this case you can't edit the displayed text.

6.4.7.4.3 Atomic AIL Editor

The AIL Editor provides a graphic subview showing the atomic function block including its interface and scrolling list subviews for the interface items (inputs and outputs).

The interface items can be selected interactively either inside the graphic subview (selecting works in a similar way as in the Composite Editor) or in one of the scrolling list subviews. All these subviews provide selection sensitive pop–up menus to edit the interface.

The main edit functions provided in the graphic subview and the scrolling list subviews are listed briefly:

* creating interface items (only inside the graphic subview)
* moving interface items (only inside the graphic subview)
* cutting interface items
* copying/pasting interface items (pasting only inside the graphic subview)
* copying the atomic function block (to a Composite Editor)
* changing the attributes of interface items (e.g. type, initial value, etc.)
* generating and printing documentation

The AIL Editor provides a text subview that can be run in inspect mode (read only mode). Usually if an interface item is selected, the text subview provides information about the selected item in read only mode.

If there is no interface item selected, the text subview shows the comment or the AIL implementation of the atomic function block. In this case the subview runs in edit mode. The user will be able to specify whether he wants to change the comment or the AIL implementation. This may be realized via related buttons (**Code** , **Comment**). The text subview provides the common text editing functions. These comprise:

* typing in text
* selecting text
* copying text (possibly cross text windows)
* cutting text

Since the text subview allows text copying cross windows it is possible to copy text from other text editors, i.e. at least this way the model developer is able to make use of other text editors (possibly file based) to implement an atomic function block.

It is possible to copy the edited block to a Composite Editor. Furthermore the AIL Editor allows to copy/paste interface items from and to a Composite Editor, another Atomic AIL Editor, an Atomic Decision Table Editor or a Composite Interface Editor.

The user may generate and print out documentation of the atomic function block. The following outputs are provided:

* textual output describing a selected i/o-item (e.g. type, initial value)
* textual output of the atomic function block's comment
* textual output of the atomic function block's AIL implementation

Copyright per DIN 34

User Interface

Figure 6–66 shows the user interface of the AIL Editor. It provides scrolling list subviews (*input list*, *output list*) and a graphic subview allowing interface item selection (*graphic subview*). A read only text subview is used to provide textual feedback to the user (*message subview*). A text subview that can be run in read only mode as well as in write mode is used to give textual information of the selected interface item (*text subview*). When there is no interface item selected, the *text subview* runs in edit mode allowing to inspect/edit the comment or the AIL implementation of the atomic function block.

Editing of the interface is done inside the *input list*, *output list* and *graphic subview* via selection sensitive pop–up menus.

> If there is no selected i/o – item, the text subview provides a text editor allowing to implement the atomic function block by AIL code.

Saving the edited text within the text subview is done by clicking on the **Apply** button, whereas via the **Reset** button one goes back to the last saved version.

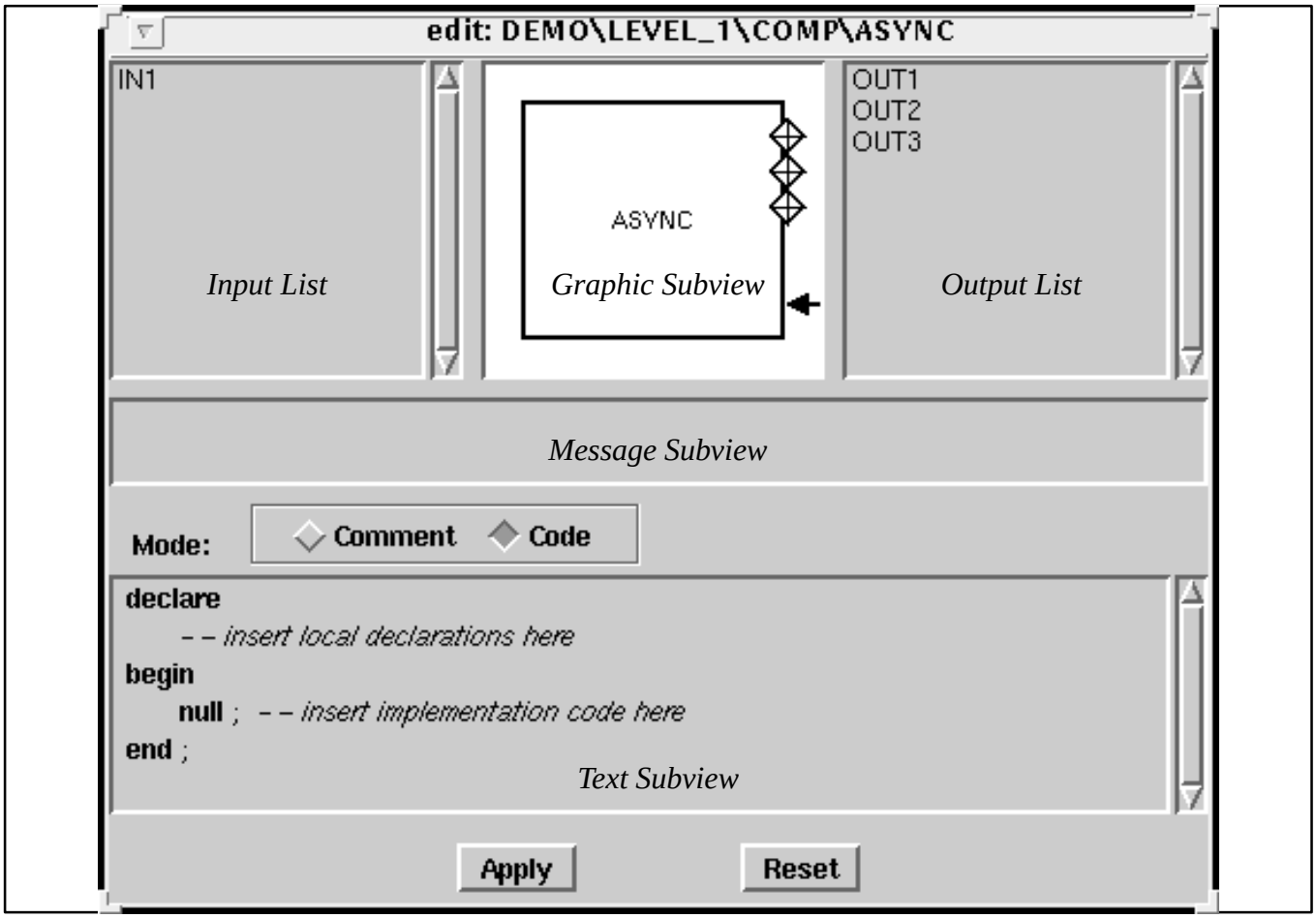


Figure 6–66 : *AIL Editor user interface*

6.4.7.4.3.1 AIL special features (hibernate, PULSE type parameter)

AIL is defined as a subset of the Ada language. (Refer to the Ada language standard for further information.) Following Ada features are out of the scope of the AIL editor:

Copyright per DIN 34

- Ada tasking
- Address Clauses

Note that the length of a line is restricted to 80 characters.

There are some special features concerning PULSE type variables and the HIBERNATE function which will be described in detail here.

Set a synchronous FB to hibernate in AIL code

- Implement several lines of code which shall be performed before the hibernate command.
- Implement the condition under which the synchronous FB shall hibernate.
- Type **hibernate := true;** in the AIL code.
- Finish the AIL code correctly.

This implementation works as follows:

- in the beginning the synchronous FB is activated cyclically
- as soon as the conditions in which the hibernate command is embedded become true the hibernate command will be performed
- after finishing the AIL code the FB is in hibernate state and needs an activating event from another FB to start its cycle again

PULSE and BURST_PULSE parameter need to be treated in a different way than all the other parameters.

How to check wether a PULSE parameter is set

- Define an input of type PULSE.
- Implement the check as follows: **if <input> = PULSE_TRIGGERED then .... end if;**
  (<input> is any name)

How to pass a PULSE parameter forward to another FB

- Define an output of type PULSE.
- Implement the conditions on which the PULSE parameter shall be passed.
- Type in the AIL code **<output> := PULSE_TRIGGERED;** (<output> is any name)
- Finish the AIL code correctly.

Note that BURST_PULSE parameter are of type integer.

How to check wether a BURST_PULSE parameter is set

- Define an input of type BURST_PULSE.
- Implement the check as follows: **if (<input> /= BURST_PULSE_NOT_TRIGGERED) then .... end if;**
  (<input> is any name)

Copyright per DIN 34

How to pass a BURST_PULSE parameter forward to another FB

- Define an output of type BURST_PULSE.
- Implement the conditions on which the BURST_PULSE parameter shall be passed.
- Type in the AIL code **<output> := <integer>;** (<output> is any name , <integer> is a number)
- *Note that <integer> determines the number of pulses in the BURST_PULSE parameter.*
- Finish the AIL code correctly.

6.4.7.4.3.2 Variables in the AIL code

Variables defined as inputs and outputs are used as variables in the AIL code too.

There are some special rules for the use of vector and matrix variables:

Each element or component must be written separately.

Example: Variable Price is a vector with 3 elements of type REAL.
Price(1) := 1.2;
Price(2) := Price(3);

Variable MAT is a matrix with 6 elements of type BOOLEAN
MAT(2,3) := TRUE;
MAT(1,1) := MAT(1,2);

Vector and matrix variables of the same type and size can be connected.

Example: Variable New_price is a vector with 3 elements of type real.
New_price := Price;

Variable MAT_X is a matrix with 6 elements of type BOOLEAN.
MAT_X := MAT;

- *Refer to the Ada reference manual for more information about operations of unconstrained array types. See also chapter 6.4.12.*

There are some special rules for the use of record variables:

- *The name of a record component consists of the record name, an underscore and the component name.*

Example: Variable Store has 2 components: much is of type REAL, enough is type BOOLEAN.
Store_much := 12398.12;
Store_enough := FALSE;

Copyright per DIN 34

- *Record variables of the same type and size can not be connected.*

  Example: Variable N_Store has 2 components: much is of type REAL, enough is type BOOLEAN.

  It is **not allowed** to write N_Store := Store.

  To connect the variables each component has to be connected separately.

  Example: N_Store_much := Store_much;
  N_Store_enough := Store_enough;

- *Note that the record components can be treated like normal input/output variables of the same data type within the AIL code. This means all operations defined for a specific data type are available for the record component.*

6.4.7.4.3.3 Restrictions

The following functions are not yet available in the current implementation.

- The necessary functions to use CSS type ”COMPLEX” are not yet implemented.

Copyright per DIN 34

6.4.7.4.4 Atomic Decision Table Editor

The Decision Table Editor provides a graphic subview showing the atomic function block including its interface and scrolling list subviews for the interface items (inputs and outputs).

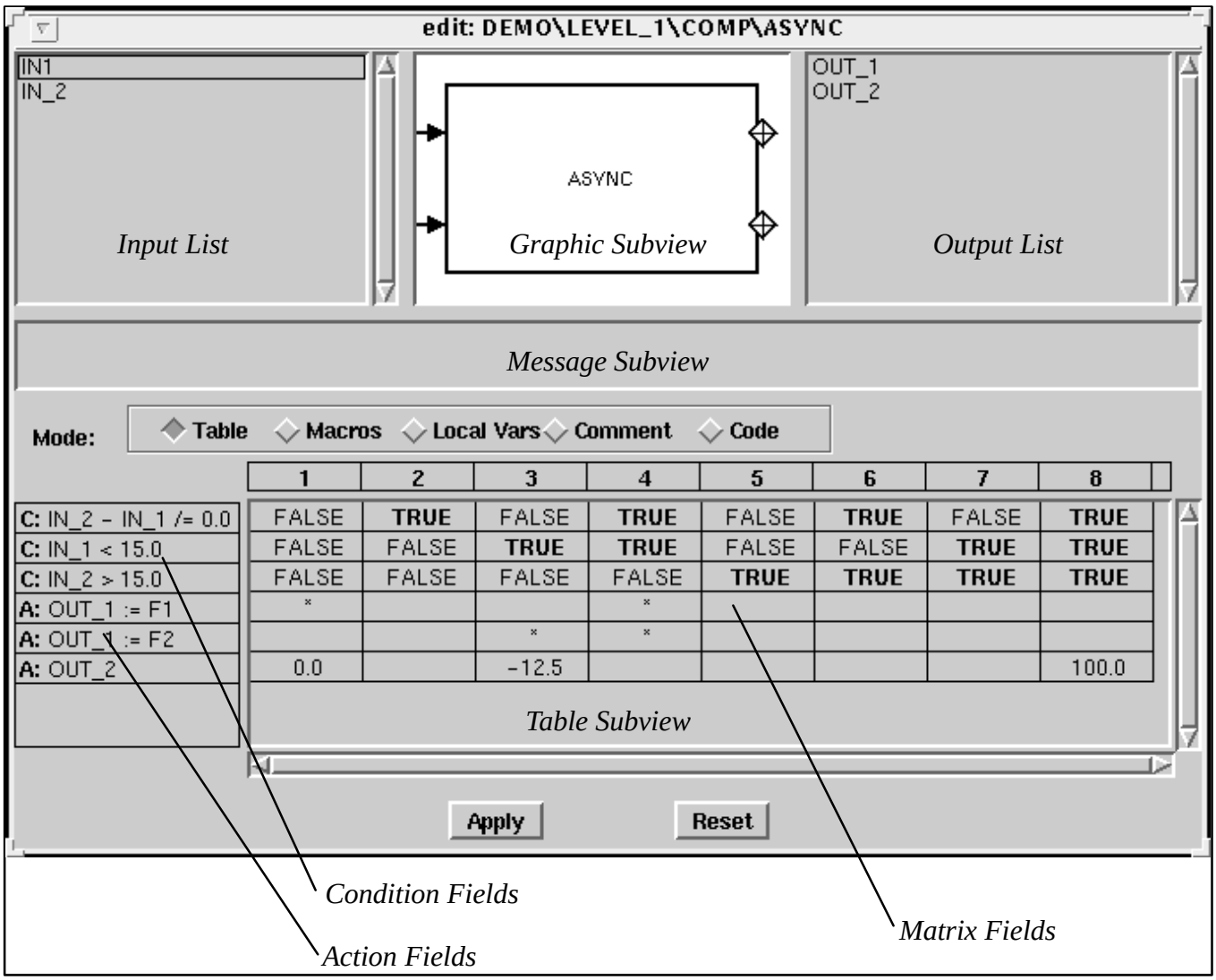


Figure 6–67 : *Decision Table Editor user interface*

The user may choose between:

* editing the decision table (table subview, **Table** button selected)
* editing macro definitions (text subview running in edit mode, **Macros** button selected)
* editing local variables (text subview running in edit mode, **LocalVars** button selected)
* editing the atomic function block's comment (text subview running in edit mode, **Comment** button selected)
* inspecting the generated AIL code (text subview running in inspect mode, **Code** button selected)

Copyright per DIN 34

* inspecting the selected interface item (text subview running in inspect mode, buttons deselected automatically)

If no interface item is selected, switching between the first five alternatives listed above is performed via related buttons.

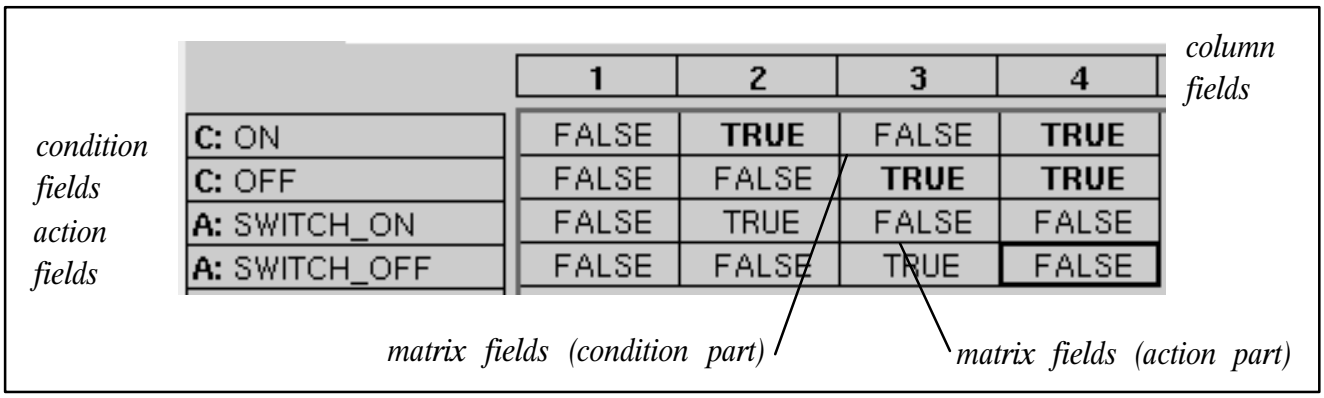


Figure 6–68 : *Structure of a decision table*

How to built a decision table

Figure 6–68 shows the general structure of a decision table. The conditions **C1:** and **C2:** concern the input variables called ON and OFF, both of type boolean. The matrix fields show all possible combinations in its condition part. As soon as a condition is added the condition part will be expanded automatically. The action part defines the actions which will be performed as soon as a certain condition is met.
Example: If the input variable ON is TRUE and the input variable OFF is false, the output variable SWITCH_ON is set to TRUE and the output variable SWITCH_OFF is set to FALSE.

There are two different ways to specify an action:

- you type the complete action in the action field and mark the desired condition with an asterisk in the matrix field or
- you specify the action incompletely, e.g. writing down the identifier of a variable that should be newly assigned. Inside the matrix fields the user has to specify either the value, an AIL expression to compute the value or a macro name.

Note that the maximum number of conditions and actions is 8.

*The labels **C1:** to **C8:** and **A1:** to **A8:** are automatically generated.*

Copyright per DIN 34

	1	2	3	OTHERS
C: INPUT = 10	**TRUE**	FALSE	FALSE	-
C: INPUT > 10	FALSE	**TRUE**	FALSE	-
C: INPUT < 10	FALSE	FALSE	**TRUE**	-
A: OUTPUT_1	C_1			
A: OUTPUT_1		C_2		
A: OUTPUT_1			C_3	
A: OUTPUT_2 := INPUT + 1	*			*
A: OUTPUT_2		123		

Figure 6–69 : *A decision table showing three different ways to fill the action fields.*

Creating a decision table

- Create an atomic function block.
- Select an atomic FB with the left mouse button.
- Press the right mouse button and select **edit –> decision table.**
- Enlarge the decision table editor window.
- Press the right mouse button and select **add condition**.
- Type the condition into the text field.
- Press the **Accept** button. The condition and the related true/false columns appear in the decision table layout.
- Repeat the previous steps until you covered all conditions.
- *Note that there are different ways to fill the action fields.*
- Press the right mouse button and select **add action**.
- Type the desired action in the text field. (see Figure 6–69, row **A4:**)
- Select the action field in the matrix with the left mouse button.
- *to be continued ....*

Copyright per DIN 34

- Press the right mouse button and select **toggle** from the pop–up menu. An asterisk appears in the selected row.
- OR
- Press the right mouse button and select **add action**.
- Type the name of an output in the text field.
- Select the action field in the matrix with the left mouse button.
- Press the right mouse button and select **edit** from the pop–up menu. Type an AIL expression, a local variable, a number or a macro into the text field. (see Figure 6–69, row **A1:,** row **A5:**)
- Press the **Accept** button.
- Press the **Apply** button to save the decision table.
- Select **Quit** from the window pop–up menu to close the decision table editor window.

Conditions and actions may be edited using macros or local variables in order to save space on the screen.

Note that you can use local variables and macros in the condition fields too.

Because the space in the different fields is limited, you can define local variables and macro definitions. The use of local variables and macros improves the readability of the decision table.

```
-- local variables

VAR_1, VAR_2, VAR_3 : UNSIGNED_INTEGER := 0;

VAR: UNSIGNED_INTEGER range 1..20 := INPUT;
```

Figure 6–70 : *Local variables definition in a decision table*

Editing Local Variables

- Press the **Local Vars** button.
- Type the definitions in the window.

Note that you have to follow the Ada syntax for the definition of local variables.

- Press the **Apply** button.
- Press the **Table** button to go back to the default decision table layout.

Editing macros

- Press the **Macros** button.
- Type the macro name, an equals sign and the macro expression in the window.

*Do **not** add a semi-colon at the end of the macro expression.*

- Press the **Apply** button.
- Press the **Table** button to go back to the default decision table layout.

Copyright per DIN 34

```
- - macro definitions

C_1 = ( ( INPUT + INPUT ) / 5 ) + 1 + VAR_1 + VAR_2 + VAR_3 + NEW_VAR * 10

C_2 = ( ( INPUT + INPUT ) / 12 ) + 2

C_3 = ( ( INPUT * INPUT ) / 2 ) + 3
```

Figure 6–71 : *Macro definitions in a decision table*

Improve the table readability

If you add a new condition, the editor completes the decision table by automatically adding all the new condition combinations to the matrix.
The matrix can be minimized in two ways: you can put an entire column to the *others column* or you dismiss only one element in a column. (see the example in Figure 6–72)

Collect entire columns to the *other columns*

- Move the mouse pointer to the column number field and select it with the left mouse button.
- Press the right mouse button and select **remove** from the pop–up menu. The selected column disappears, a new column *others* appears in the matrix field if there is not already one.

There is a way to group columns which differ only in one condition which will not be used.

Group two columns to one

- Select a field in the condition part of the matrix with the left mouse button.
- Press the right mouse button and select **no matter** from the pop–up menu. The TRUE/FALSE is replaced by a # sign.

The decision table can be set to its original layout at any time, select the *others* or # field and perform the **expand** command.

Copyright per DIN 34

	1	
c1: in1	**true**	**others**
c2: in2	**true**	
a1: out1	true	false

minimized by others column

incomplete AIL statement

	1	2	3
c1: in1	#	**false**	**true**
c2: in2	**false**	**true**	**true**
a1: out1 := true;			*
a2: out1 := false;	*	*	

Figure 6–72 : *Logical AND gate implemented by decision table (2 examples)*

*Select the **reset** button and you go back to the last saved version.*

Copyright per DIN 34

6.4.7.5 Icon Editor

6.4.7.5.1 Basics

The Icon Editor runs in a separate MDE window. It may be activated from within the Composite Editor. The bitmap based black and white Icon Editor allows to inspect and edit the icons of composite and atomic function blocks.

Easy to understand iconized buttons allow the user to choose the kind of graphic operation he wants to execute. He may choose from:

* setting a single pixel
* drawing a line
* drawing a rectangle
* drawing a circle
* placing text into the icon
* selecting and moving a part of the icon’s bitmap
* selecting and copying a part of the icon’s bitmap

The editing operations are performed inside the icon edit subview via mouse. The edit subview shows a magnified copy of the original icon. It provides vertical and horizontal scrolling.

The user can specify the colour for setting single pixels, drawing lines, rectangles, circles and placing text. He may choose between black and white. The icon edit subview provides two modes for drawing rectangles and circles allowing to draw the border of the object or to draw a filled representation.

Optionally the user can decide to display a grid raster inside the edit subview. It is possible to clear the entire icon (set all pixels to white) and to invert the icon.

The current colour (black or white) and mode (border or fill) are shown in the Icon Editor’s status subview. Additionally the status subview shows the current text specified for the text placement operation.

Another non–interactive subview shows the icon in its original size.

User Interface

Figure 6–73 shows the user interface of the Icon Editor. It provides an icon edit subview. A bar of iconized buttons allows the user to select the current graphic operation. A non–interactive view displays the edited icon in its original size. A number of buttons and a text input field are used to show the current status.

Copyright per DIN 34

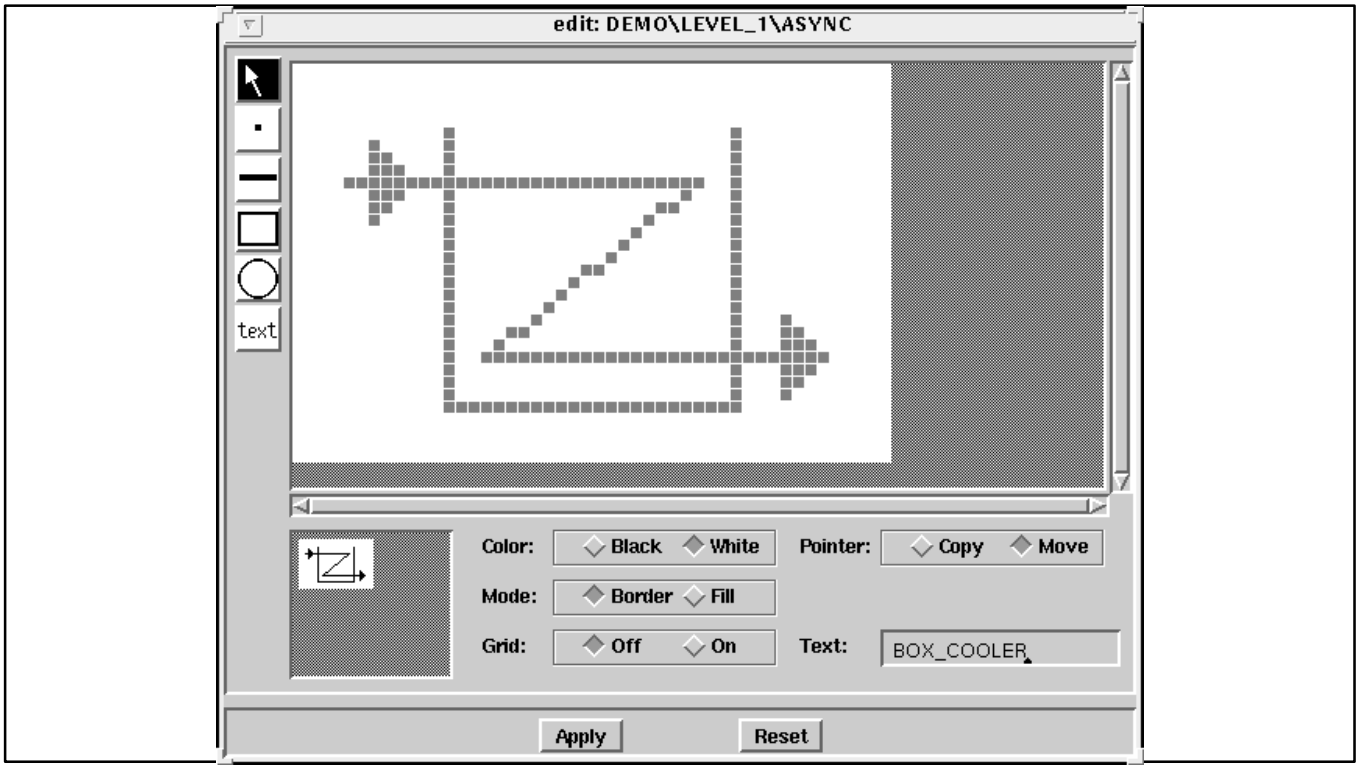


Figure 6–73 : *Icon Editor user interface*

6.4.7.5.2 Components

The Label

The label of the Icon Editor shows the pathname of the function block.

The Edit Subview

The edit subview is the place where icon editing actually is performed. It shows a magnified view of the edited icon. In general, drawing is performed by pressing and releasing the left or middle mouse button.

The Tool Buttons

By activating the different tool buttons, you specify what kind of object you want draw inside the edit subview. You can choose from pixel, line, rectangle, circle and text.

If the pointer button is activated, you can move resp. copy parts of the edited icon. To do so, press and hold down the left mouse button. A flashing rectangular outline appears on the screen. Move the cursor around and designate the area of the icon you want to move or copy, then release the button. Press and hold down the left mouse button again while moving the cursor around. The selected icon part appears highlighted following the cursor. Move it to the desired position and release the mouse button.

The Icon Subwindow

Copyright per DIN 34

The icon subwindow shows the edited icon in its original size.

The Status Area

The status area shows the Icon Editor's current status.

Color indicates the color that is used to display the objects you create inside the edit subview. The color can be set to *black* or *white*.

Mode indicates the mode that is used to display rectangles and circles. You can choose between *border* and *fill*.

Grid indicates whether inside the edit subview a grid raster should be displayed or not.

Text indicates the text you can create when the text tool button is activated.

Color, mode and grid can be set by activating the corresponding button. The text input field allows to type in the text to be created.

Working with the Icon Editor

You open the icon editor on a selected composite or atomic function block inside the Composite Editor. Initially the blocks are labeled with their name.

Create an icon and choose the menu command **Apply**. Note that on executing **Apply**, the selected function block inside the composite editor becomes labeled with the specified icon.

To remove the icon from a function block, select the function block inside the Composite Editor and execute the menu command **icon–>cut**. The icon disappears from the function block, it is labeled with its name again.

Copyright per DIN 34

6.4.7.6 Tree Browser

The Tree Browser tool can be called within the Database Browser and on any level of the model hierarchy.

Note : Since the collection of the data required for the Tree Browser display, especially for a large model, may take quite some time, the user is prompted for confirmation before the Tree Browser is actually started.

After selection of a function block the user can jump via edit/inspect commands to the related hierarchy level.

The tree browser provides the same documentation generation function (command **print**) as the composite editor resp. inspector (see the related chapters).

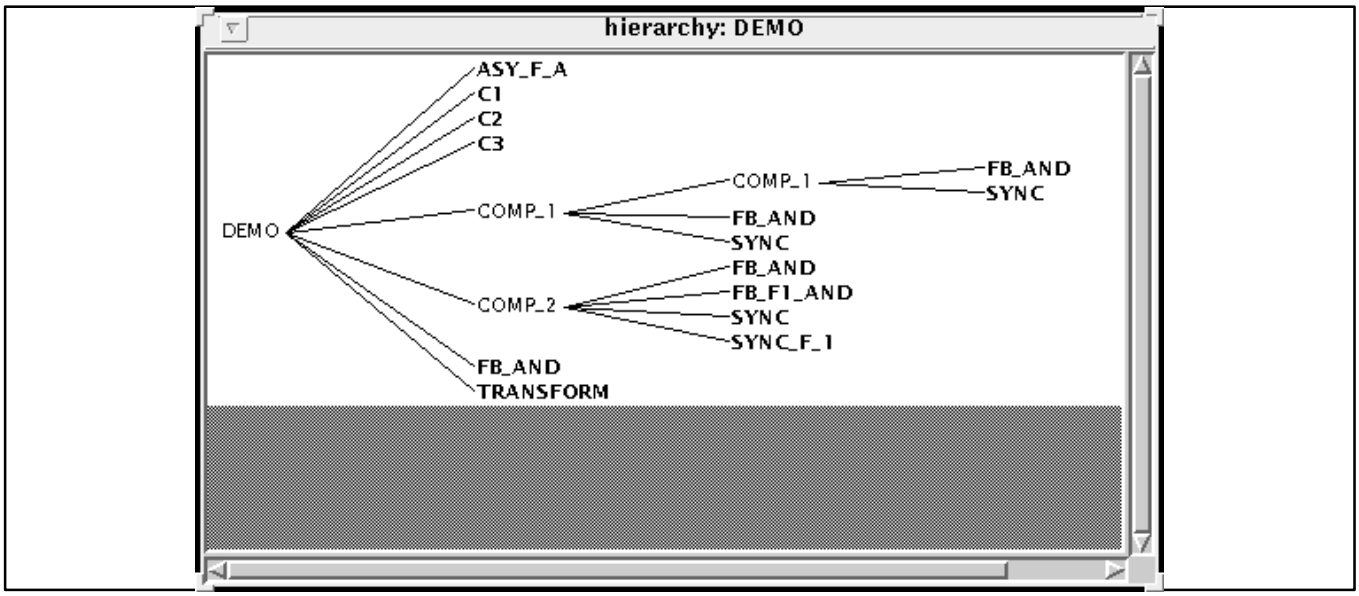


Figure 6–74 : *The Tree Browser user interface*

Copyright per DIN 34

Navigate in the model tree structure

- Select **goto** in the Composite Editor's basic menu. The tree browser window (see Figure 6–74) appears.
- Scroll to the desired part of the model.
- *Note that atomic function blocks and parameter blocks are written in **bold** style.*
- Click on the desired item. The item is highlighted.
- *Perform the next two steps if the selected item is a composite FB.*
- Press the right mouse button and select **edit–>implementation** from the pop–up menu. The Composite Editor switches its currrent level and shows the selected composite FB's inside view in overview mode.
- Select **edit** from pop–up menu.

  OR

- *Perform the next step if the selected item is an atomic FB.*
- Press the right mouse button and select **edit–>AIL** or **edit–>decision table** from the pop–up menu. An atomic editor window appears.
- Select **Quit** from the window menu of the tree browser window.

Copyright per DIN 34

6.4.7.7 Documentation Generation Function

Note that there are several ways to get model print–outs.

The Documentation Generation Function runs in a separate MDE window. It may **only** be activated from within the Database Browser.
Refer to section 6.4.7.1.41 where is described how to get a listing while you are inside a model. Note that every editor provides a separate printing function.

The Documentation Generation Function provides a graphic subview showing a vertically and horizontally scrollable tree view that allows to select the desired parts of the model by clicking the mouse on the item.

The user can choose between a print on a laser printer or the creation of a text document which can be handled by the CGS SDE text system Interleaf.

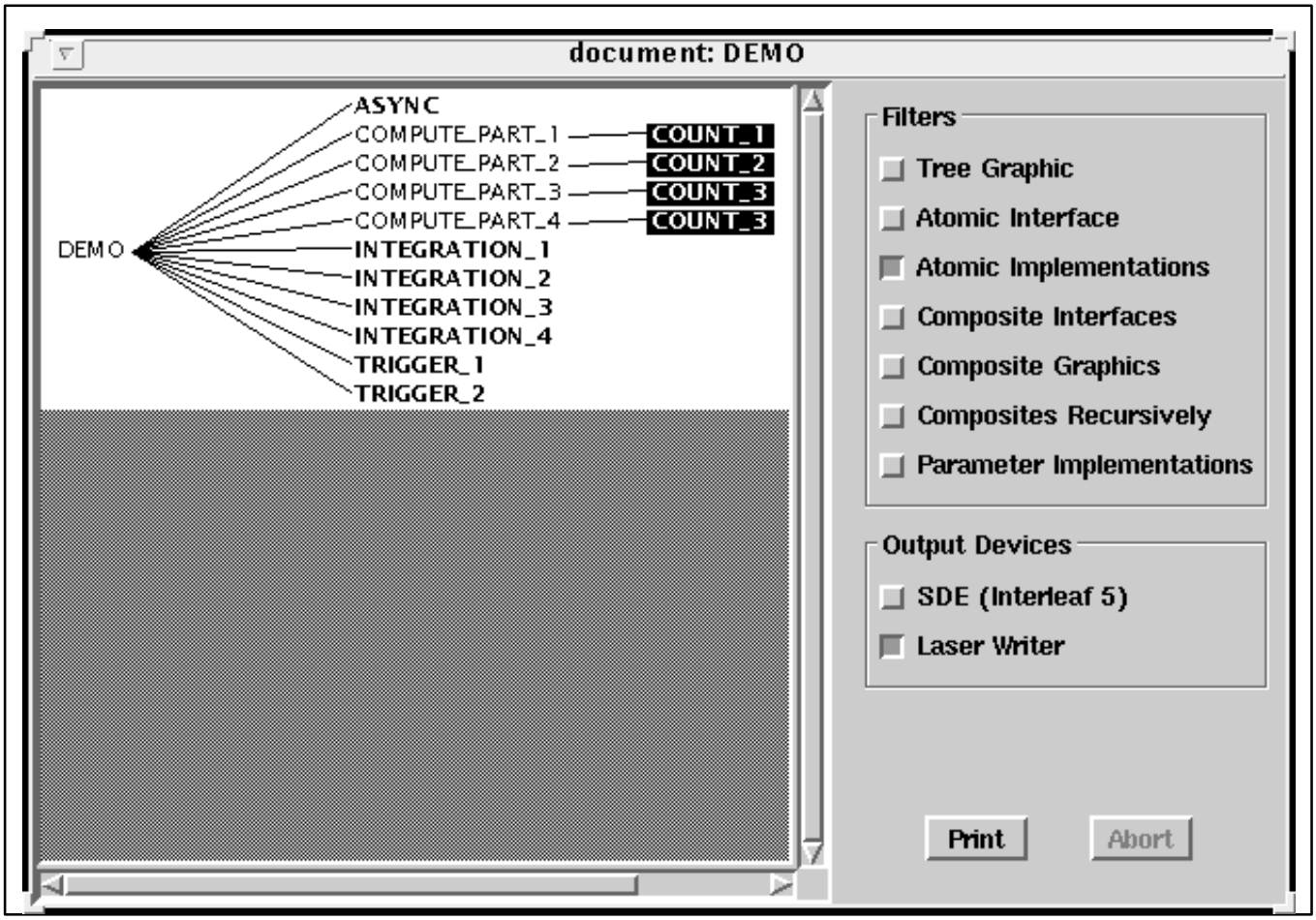


Figure 6–75 : *Printing the contents of the atomic FBs COUNT_1 to COUNT_4*

Copyright per DIN 34

Printing model documentation

- Open the Database Browser window.
- Select the model by clicking on its name with the left mouse button.
- Press and hold down the right mouse button.
- Move the mouse pointer to the **print** command and release the mouse button. The documentation generation window (see Figure 6–75) appears.
- Use the scroll bars to display the desired part of the model.

*Note that the **Tree Graphic** Filter is the only item which needs no selection in the displayed model tree structure (tree subview).*

- Click on the desired function blocks in the model tree structure (tree subview). The names become highlighted.
- Select the contents of the printout in the **Filters** list.
- Select the **Output Device**. Click on either the **Laser Writer** button or (to get a text document) the **SDE** button.
- Press the **Print** button.
- Select a printer from the pop–up list of available printers.

*Note that the contents of the printer list depends on the installation environment. If there is only one printer available a field **default** appears.*

To create the model description as an Interleaf document, follow the steps in the procedure. Select **SDE** as the output device. Automatically a book with the models name will be created in your desktop directory.

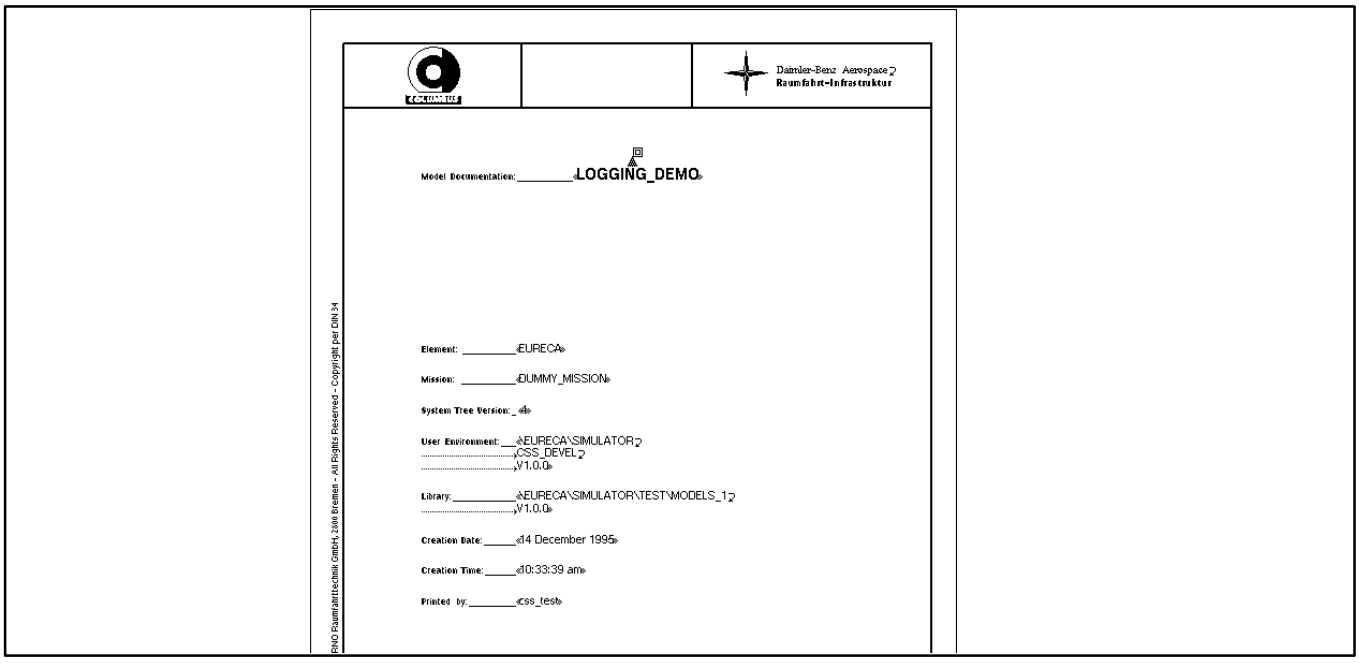


Figure 6–76 : *The cover page of the model document*

Modifying the default model documentation layout

- Invoke the documentation tool.
- Open the book generated by CSS (the name of the book is equal to the model name).
- Open and change the catalogs Definition and/or Main_Catalog.
- Save your changes.

The modification of the model document layout for all CSS users is a task which can be performed only by a priviledged user (e.g. system administration). The standard book layout is located in the $CSS_HOME/ config/ileaf directory. Copy the book CSS.boo to your desktop, make the required changes and copy it back. From now on all CSS users will get the new layout in the model documentation.

Note that pictures taken from the model are not visible in the Interleaf document. Nevertheless they appear as soon as the page is printed.

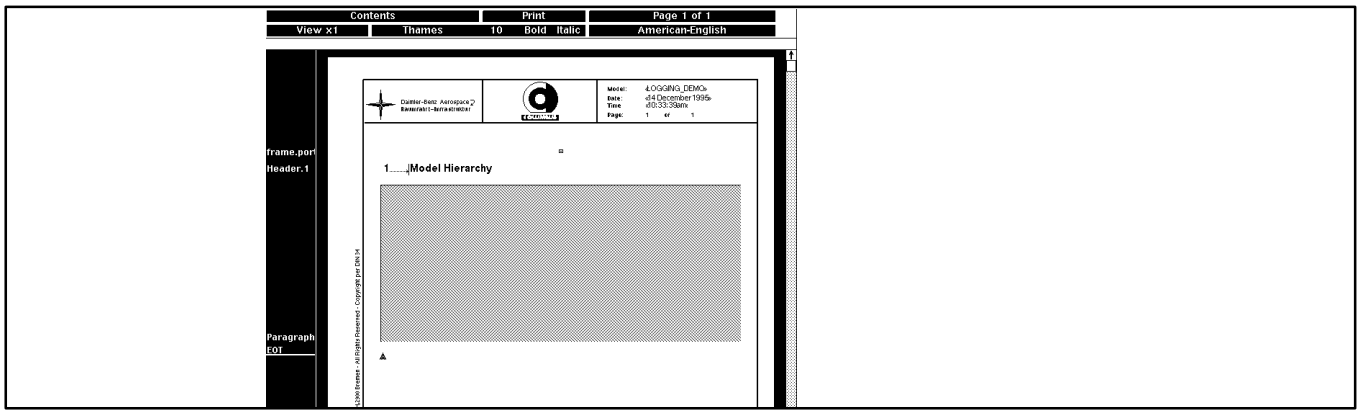


Figure 6–77 : *This grey rectangle shows the model hierarchy after printing*

Copyright per DIN 34

6.4.7.8 Onboard References Adaptation Function

6.4.7.8.1 Basics

Each interface item of a database based simulation model may reference an onboard end item contained in the database, i.e. it may keep an onboard reference which is actually the database pathname of an onboard item. When a database based simulation model is loaded, its onboard references are checked on validity (accessability of the referenced onboard items in the current CCU scope and type compatibility, if appropriate). All onboard references that have become undefined and are not adapted, will be reset automatically, i.e. the onboard references are cleared and therefore lost.

An onboard reference is said to be undefined, if an onboard item with the given pathname cannot be accessed (is not visible) in the current CCU scope. There may be a large number of undefined onboard references if a model is loaded under the wrong CCU by mistake. In such a case, the windows opened on the model should be closed without saving the model, and then the model can be reloaded under the correct CCU.

However, it may also be the case that individual onboard items have been renamed, removed and substituted by other onboard items, or that entire subtrees have been moved to another location in the database. In order to avoid the need for setting once again all these onboard references one by one using the Composite Editor resp. Composite Interface Editor in conjunction with the Database Browser, CSS provides the Onboard References Adaptation Function which facilitates the task of adapting the simulation model's onboard references.

The Onboard References Adaptation Function runs in a separate MDE dialog window. To activate it, make sure that the simulation model is not loaded, i.e. there may be no open window on the model. Then select the simulation model in the item subview of the Database Browser and choose one of the commands **Edit->Implementation** or **Edit->Interface** in the pop-up menu. If the model contains undefined onboard references, the dialog window of the Onboard References Adaptation Function appears on the screen allowing to adapt these onboard references before the resp. editor window is opened; otherwise the Composite Editor resp. Composite Interface Editor window is opened directly.

User Interface

Figure 6–78 shows the user interface of the Onboard References Adaptation Function. It provides two scrolling list subviews showing the undefined and adapted onboard references. Two text input fields and a number of buttons allow adapting of onboard references. A read only text subview is used to provide textual feedback to the user.

Working with the Onboard References Adaptation Function

The upper scrolling list subview labeled **Undefined Onboard References** provides the list of yet undefined onboard references, the lower one labeled **Adapted Onboard References** shows the list of meanwhile adapted onboard references. Both subviews allow multiple selection. Below each subview, if exactly one of the listed onboard references is selected, the name of the referencing model interface item (i.e. top level I/O) is given.

Copyright per DIN 34

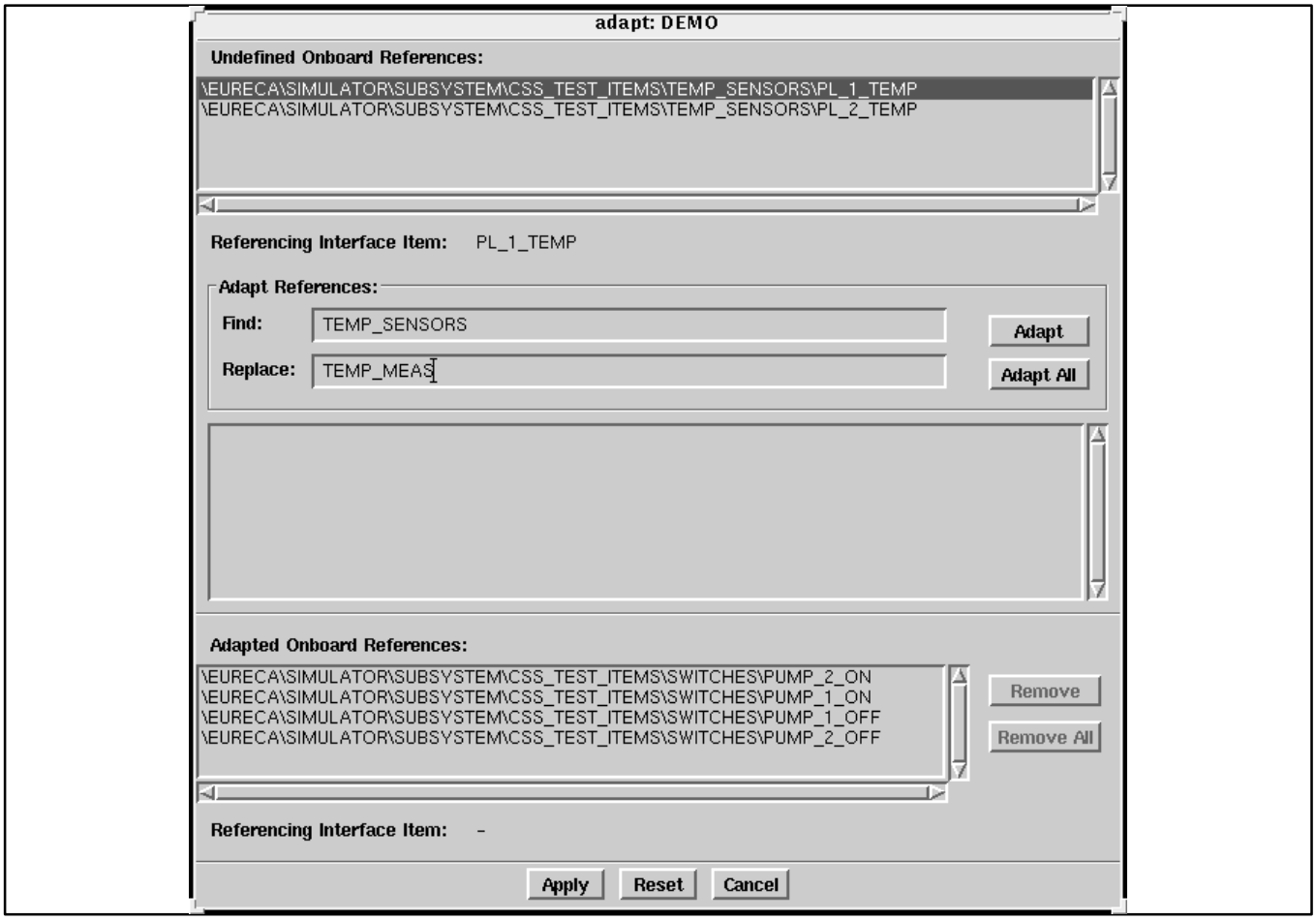


Figure 6–78 : *The Onboard References Adaptation window*

Adaptation is performed using the two text input fields. In an undefined onboard reference, the first occurence of the characters typed in the text input field labeled **Find** is searched for, and if found, is replaced with the characters typed in the text input field labeled **Replace**. Then an attempt to set the onboard reference to this new generated pathname is made, and if successful, the undefined onboard reference is removed from the upper scrolling list subview and the new onboard reference appears in the lower scrolling list subview. An attempt that failed, e.g. because an end item could not be located in the current CCU scope or the item found is already referenced by another model interface item, is indicated in the message subview (the read only text subview).

Pressing the **AdaptAll** button applies adaptation to all undefined onboard references provided in the upper scrolling list subview while pressing the **Adapt** button restricts adaptation to the undefined onboard references currently selected in this subview.

The **Remove** resp. **RemoveAll** buttons allow to remove the selected resp. all already adapted onboard references from the lower scrolling list subview.

Pressing the **Apply** button completes the adaptation process by setting all adapted onboard references shown in the lower scrolling list subview. Pressing the **Cancel** button aborts the adaptation process without adapting any onboard reference. The **Reset** button allows to discard any adaptations performed so far and to restart the adaptation process from the beginning.

Copyright per DIN 34

6.4.7.9 The Simulation Table Editor Window

6.4.7.9.1 Creating a Simulation Table

The Simulation Table Editor runs in a separate window. It may be activated from within the DBB window. The tool allows to define the monitoring, logging or tracing elements needed to observe the behavior of dynamical values during simulation.

Creating a Simulation Table

- Open the Database Browser window. (see Figure 6–79)
- Select a model by clicking on its name with the left mouse button. The model name becomes highlighted.
- Press the **Simulation Tables** button with the left mouse button.
- Move the mouse pointer into the Info Subview II (lower right field) of the DBB.
- Press and hold the right mouse button. Then move the mouse pointer to the **add table** entry in the pop–up menu. Release the mouse button. An input window appears.
- Type a new table name (a default name is already given) in the input field.
- Press the **Apply** button. The table name appears in a list in Info Subview II.

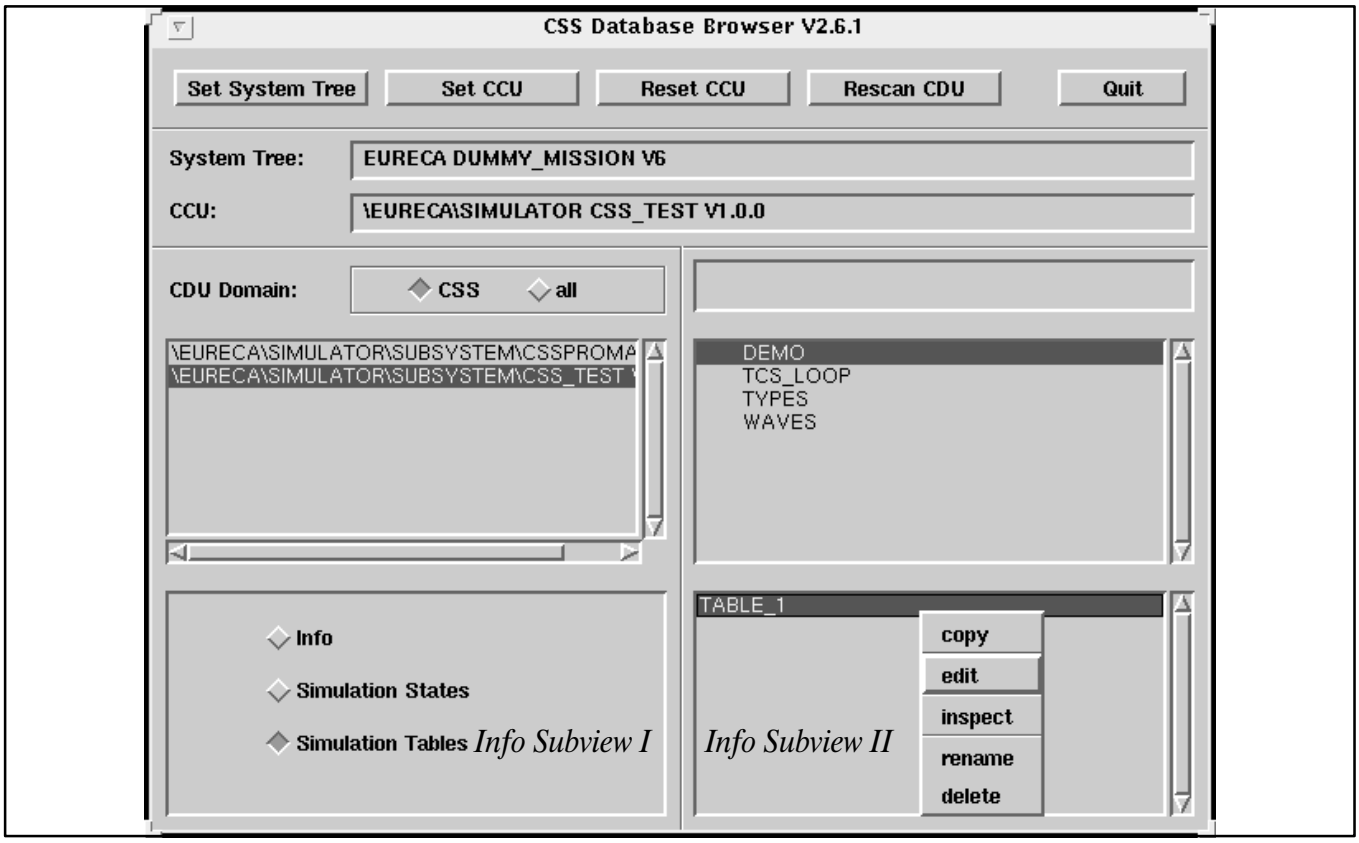


Figure 6–79 : *The Simulation table TABLE_1 is selected in the DBB window for editing.*

The Simulation Table Editor is a tool that allows to define model items for monitoring , logging and tracing.

Copyright per DIN 34

* Monitoring means to observe online dynamic changes of the model parameters during simulation execution.
* Logging means to store the change of dynamic values for offline evaluation.
* Tracing means to observe the moment of activation (execution) of functions blocks on the screen, or to record the moment of activation in a log file.

The simulation table editor window in Figure 6–80 shows on the upper left side the name of the currently edited table with the table status in parantheses. The default table status is 'unchanged', while editing the table the table status changes to 'changed'.

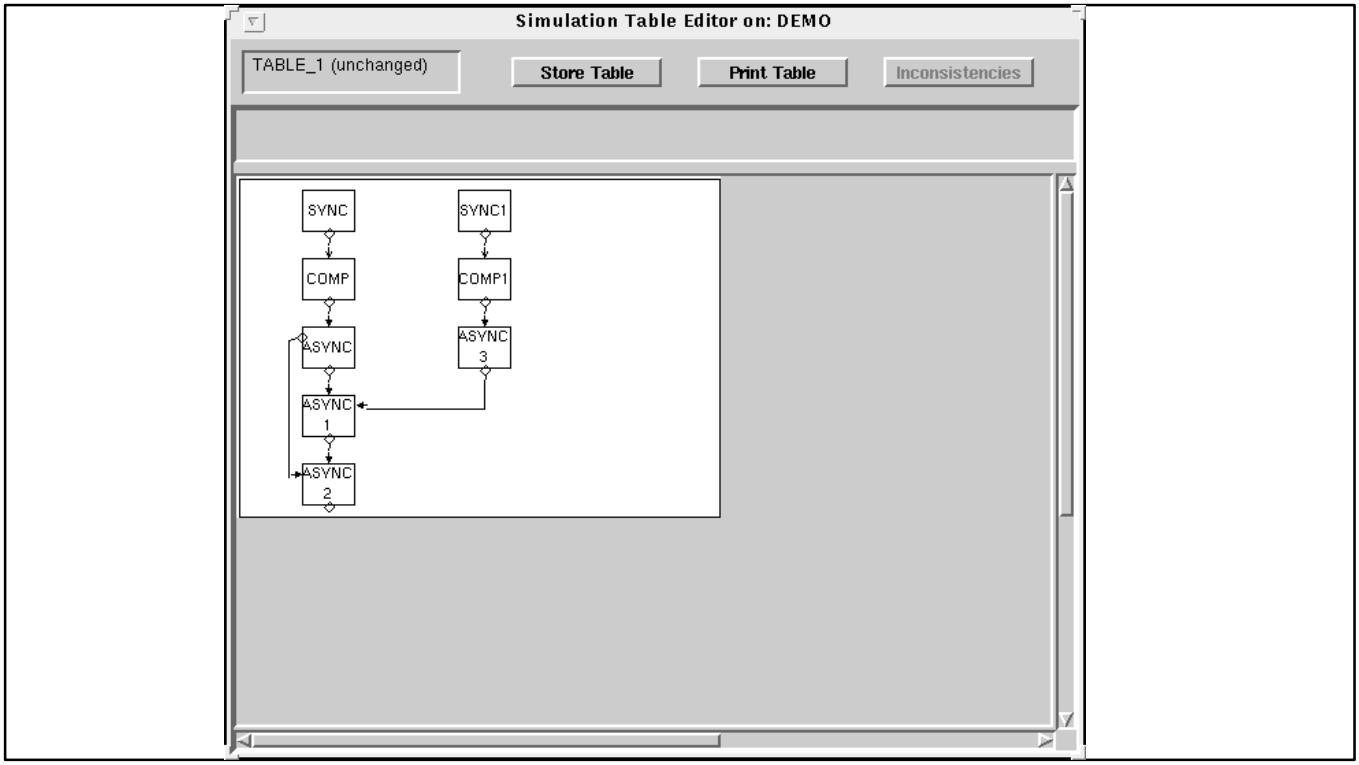


Figure 6–80 : *The Simulation Table editor window (overview mode)*

The three buttons in the upper field have the following functions:

* the **Store Table** button
stores the monitoring table. If you wish to create a new table instead of changing the old one change the name in the appearing input window.
* the **Print Table** button
prints the contents of the simulation table as ASCII listing, e. g. the complete item name, the activation status, the selected type of monitoring element, logging and tracing items.
* the **Inconsistencies** button
is normally disabled. If you load a simulation table which does not fit to the currently selected model this button becomes active. A list of inconsistencies between the model and the selected table will be displayed.

Copyright per DIN 34

- *Note that it is possible to have more than one open simulation table during model execution on the screen. In models with several levels the user may observe the changes on different levels in parallel.*
- *Due to the fact that there is no* switch on/off *feature for tracing (the tracing starts as soon as a table containing tracing items is loaded and the model execution is started) the user should split the functions, i.e. the user may decide to put all monitoring and logging items in one table or split them into several tables, but tracing items should be in a separate table in any case.*

6.4.7.9.2 Creating monitoring items in the simulation table

Prerequisite for the monitoring of dynamic values is to assign a value monitoring element to the desired I/O item.

In the Simulation Table the graphical monitoring elements appear on the background of a graphical model structure, just the same representation as for MDE.

In order to enter new monitoring definitions, the user can navigate through the model structure and select the desired item in the model's graphical representation.

The provided value representation types for graphical monitoring are:

* literal representation
* gauge representation (horizontal bar, vertical bar and dial)
* curve representation
* discrete state representation

- *Editing a table for monitoring means to assign a literal, gauge, curve or discrete state representation element to the desired inputs or outputs.*

Copyright per DIN 34

Editing monitoring elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired I/O item and select **monitor–>literally** (or any other type of monitoring element) from the pop–up menu. An input window for monitoring parameters pops up. (see Figure 6–81)
- Click in the **on change** box in the monitoring parameter window.
- *It is recommended to use the **on change** monitoring option if possible.*

  OR
- Click in the **cyclic** box in the monitoring parameter window.
- Type a number between 1 .. n in the Cycle field. The value will be monitored every nth frame.
- Press the **Apply** button. A monitoring element is displayed next to the selected I/O item.
- Click on the monitoring element and resize or move it conveniently.
- Repeat the preceding four steps until all desired items are monitored.
- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

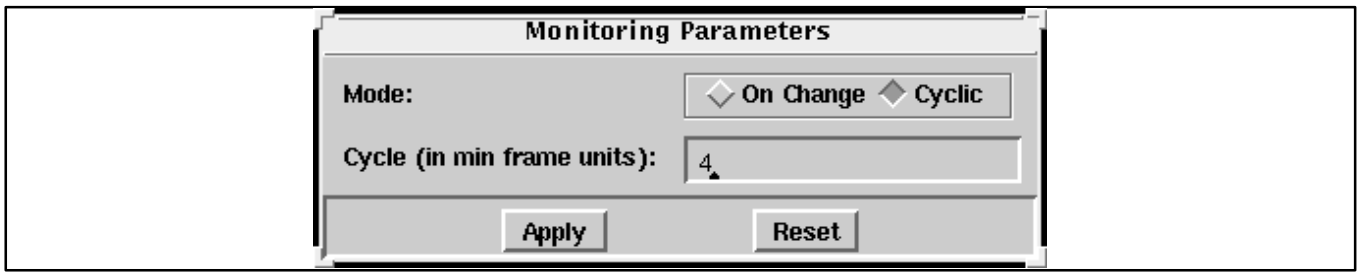


Figure 6–81 : *The Monitoring Parameters window*

On change means that the monitoring value will be displayed each time the value changes. During simulation run the monitoring table display update rate is once a second.
Note that for values changing faster than once a second you can’t observe each value during continuous simulation. Choose single step simulation mode for proper monitoring.

Copyright per DIN 34

If you select a **cyclic** monitoring note that the minimum cycle time is not limited but the same display restrictions appear.

In the preceding procedure it is recommended to use the **on change** monitoring. This is valid for parameters which do not change too often. If a parameter changes very rapidly it can be advantageous to select the **cyclic** monitoring.

In the preceding procedure only one type of monitoring element is mentioned. Literally monitoring means that the value is displayed as a string.
Depending on the type of value you want to monitor you get a list of possible graphical elements.

The provided value representation types for graphical monitoring are shown in Figure 6–82:

* literal representation
  The value is displayed as a textual string,
  this representation is possible for all value types.
* gauge representation
  The value is displayed either as horizontal or vertical bar graph or in a dial display,
  this representation is possible for all numerical types incl. duration values.
* discrete state representation
  Discrete state representation is applicable for items with a small set of alternative values, the values are displayed as a list, the current value is shown by setting a marker,
  this representation is available for boolean and state_code values.
* curve representation
  The display shows the temporal behaviour of a value. At defined steps the value is shown, not overwriting the preceding value. This representation is applicable to numerical values.

Note that monitoring can only be performed for scalar items (not allowed for vector, matrix etc data types). Use the snapshot feature during simulation execution instead.

Copyright per DIN 34

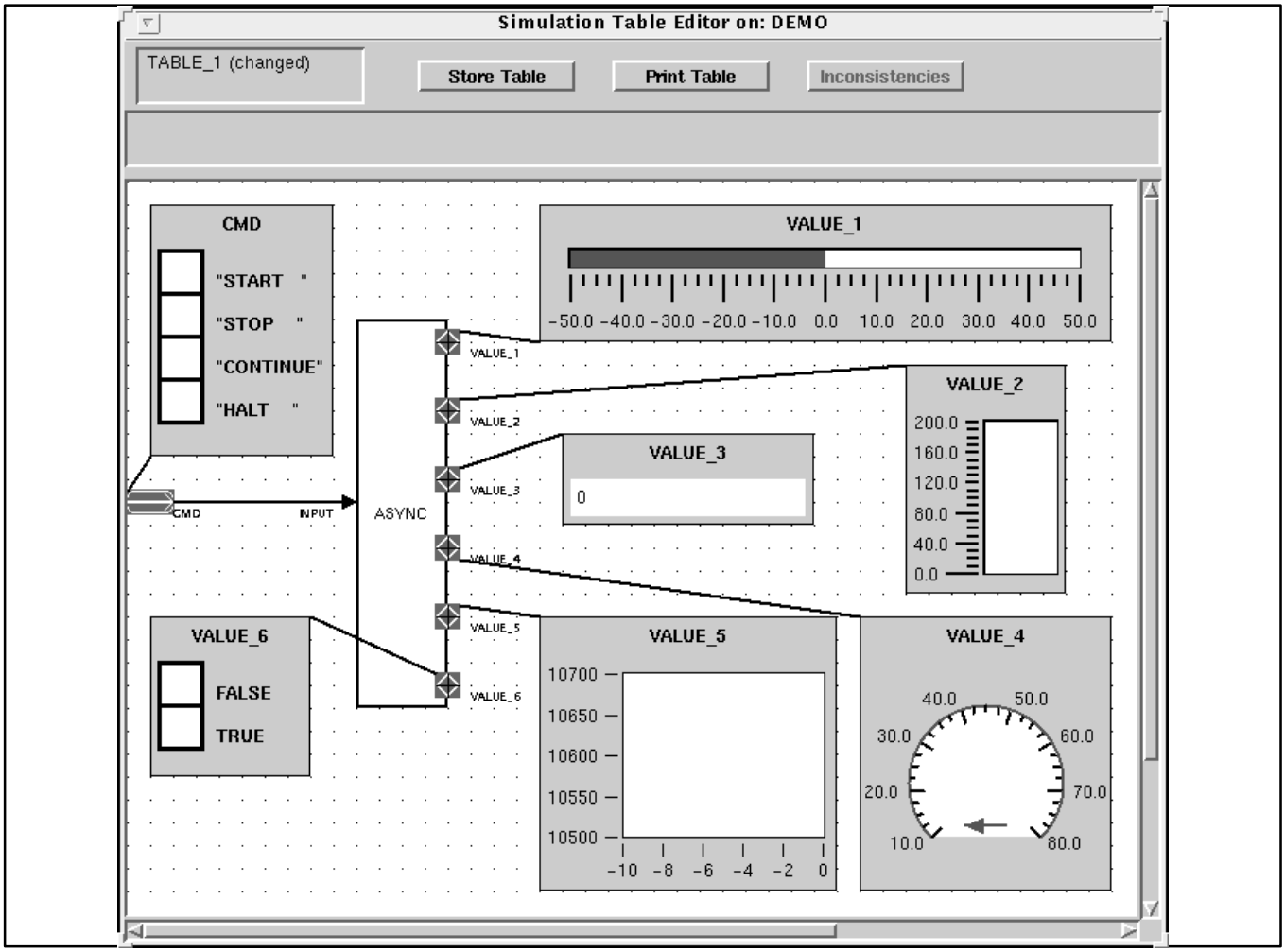


Figure 6–82 : *The graphical representation elements*

Gauge elements have a default scaling which can be changed easily.

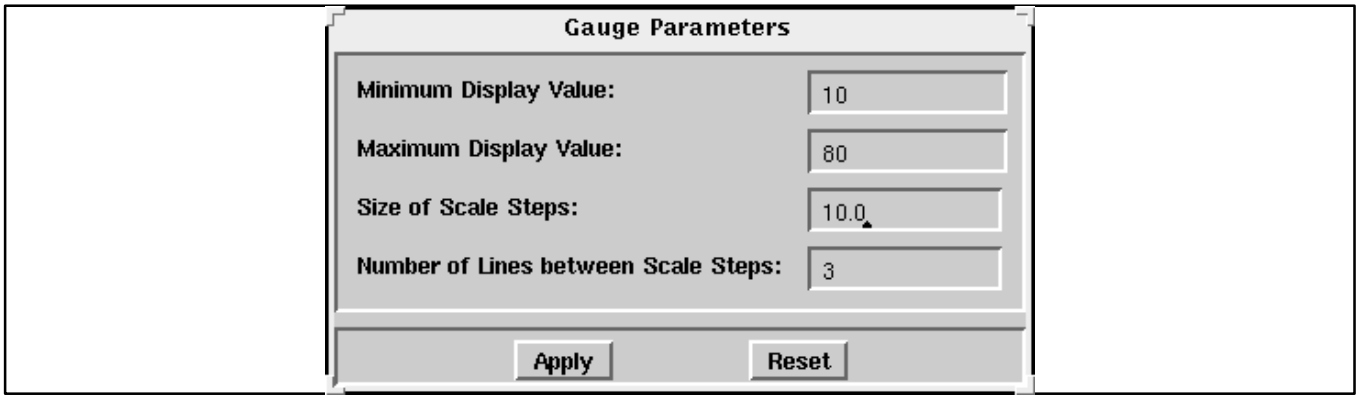


Figure 6–83 : *The gauge element scaling parameters*

Minimum and maximum display value define the range of the scale. The size of scale steps determines the distance between the numbering and the precision of scale lettering. If you write the number with a decimal point and digits behind the decimal point the lettering will

Copyright per DIN 34

be in this format.
Example: display a value between 0 and 10. Size of scale steps shall be two. The scale lettering is 0,2,4,6,8,10. If you select a scale steps size of 5, you see the lettering 0,5,10 on your scale.

The number of lines between the scale steps determines the subdivision of the scale.

Changing the scaling of gauge monitoring elements

- Click on the monitoring elements with the left or middle mouse button. The monitoring elements appear selected.
- Press and hold the right mouse button. Select **parameters** from the pop–up menu.
- The Gauge Parameters window (see Figure 6–83) appears.
- Change the parameters accordingly.
- Press the **Apply** button.
- Check the display and repeat the preceding steps if necessary.

You can delete monitoring elements either by selection of the element and **remove** from the menu or by clicking on the I/O–item and **remove definitions.** The latter command removes all monitoring elements connected to the I/O item.

The other commands available within the simulation table editor window do not differ from the commands in the MDE windows described in previous chapters.

6.4.7.9.3 Restrictions on monitoring

Monitoring elements can be assigned to all atomic and top–level inputs/outputs.
A different point are the borders of a composite function block. Inputs and outputs can be monitored either within the composite function block or outside, but not on both levels.

As Figure Figure 6–84 shows it is not possible to assign a monitoring element to the input on the left side of the composite function block. The inputs to a composite function block can be monitored only on the next lower level (inside view).

On the other hand it is not possible to monitor the output on the inner right side of the composite (see Figure Figure 6–84 – inside view). Outputs to the next higher level can be monitored on the higher level only (the outside view shows the monitoring element assigned to the output).

Copyright per DIN 34

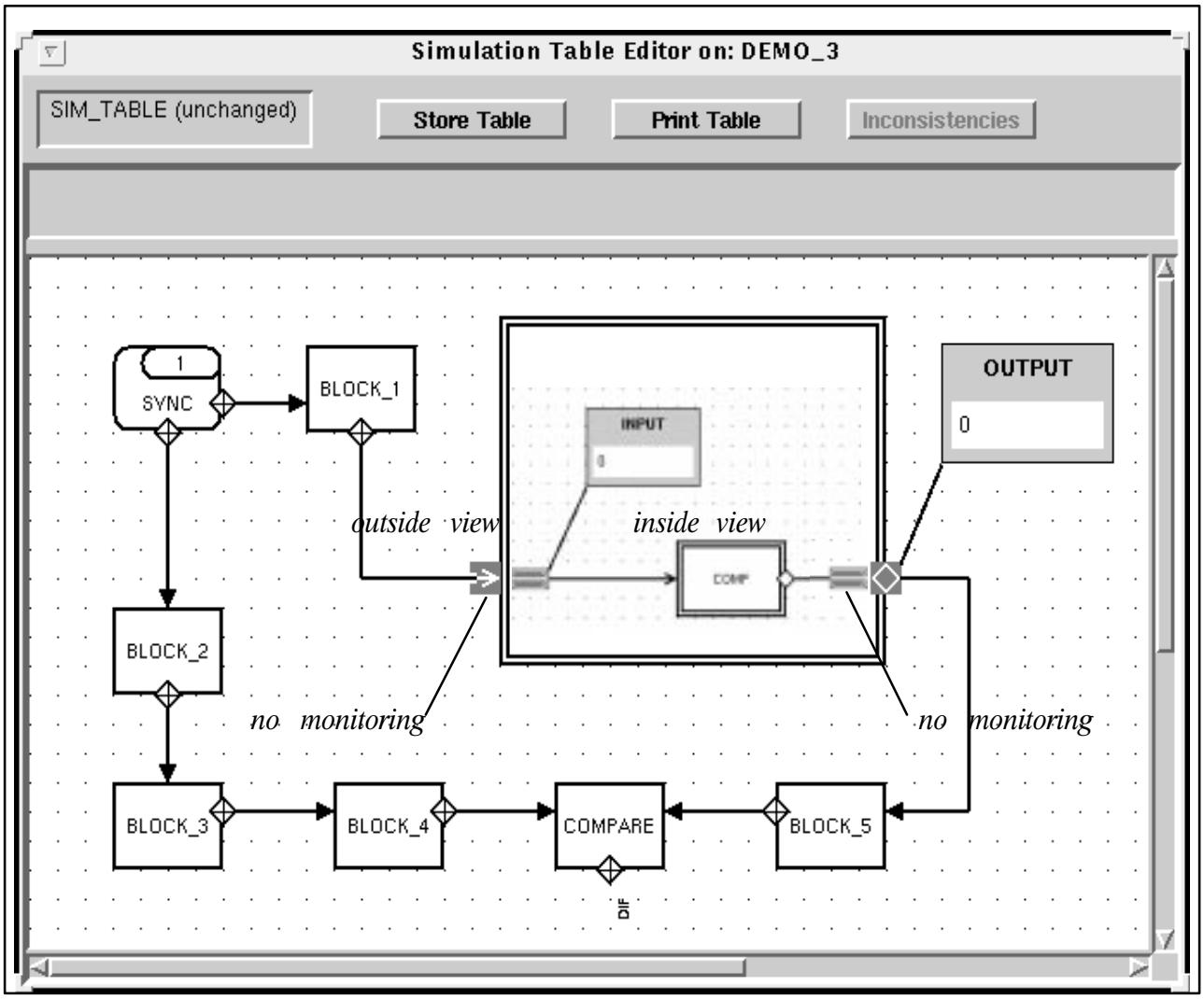


Figure 6–84 : *Monitoring I/O items*

I/O items at the border of a composite function block monitored on a level which is currently not visible are marked with a grey underlay.

6.4.7.9.4 Creating logging items in the simulation table

An interface item can be logged **on change** (i.e. the value is written each time the value changes) or **cyclically** each minframe. The items marked for logging are displayed in the simulation table editor window with a grey underlay.

The logging results are stored in *archive files* (ASCII text format) for post simulation evaluation. Each entry listed comprises the logged item's RID, the variable type, a value and the time (local time and simulated mission time) the value was logged.

In addition, it is possible to store the results in *data set files* (binary format) which can be processed by the Test Evaluation S/W (TEV); for more information on data set processing refer to the TEV user manual.

Copyright per DIN 34

Editing logging elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired I/O item and select **log–>on change to archive** resp. **log–>on change to archive and data set** from the pop–up menu. The I/O item is shaded with grey. (see Figure 6–85)

  OR

- Click on the desired I/O item and select **log–>cyclically to archive** resp. **log–>cyclically to archive and data set** from the pop–up menu. The I/O item is shaded with grey.

Note that **log–>cyclically** *means that the values are recorded every min frame.*
The recommendation is to use the ***on change*** *option.*

- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

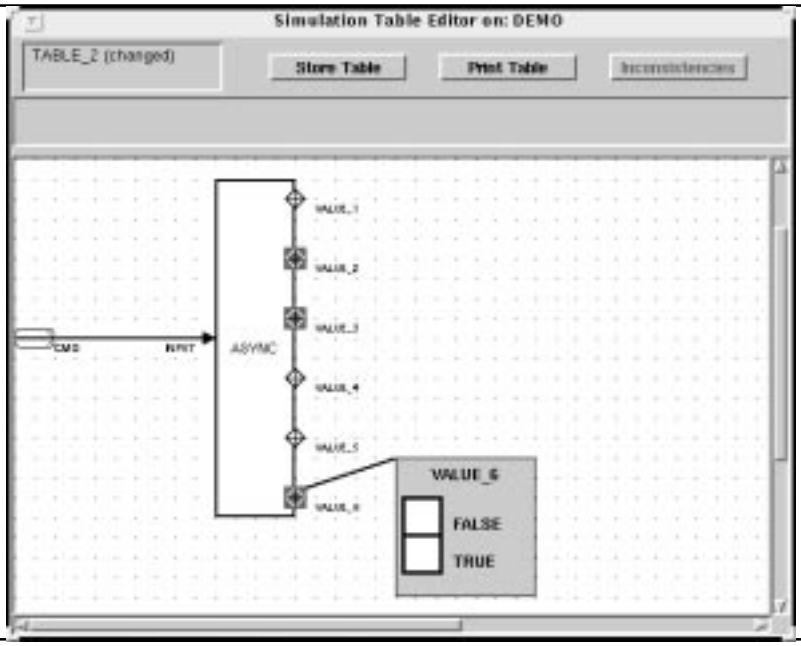


Figure 6–85 : *The outputs VALUE_2, VALUE_3 are logged*

Copyright per DIN 34

The files containing the logging results (i.e. the *archive files* and optionally the *data set info files* and *data set files*) will be stored as soon as the session owner finishes the simulation session (i.e. the simulator kernel is stopped) or a user explicitely presses the button **Logging: Store Results** in the Simulation Controller; this is possible repeatedly while the simulation is running. Each time the logging results are stored, new files are created comprising the data accumulated since the last storing or since the start of the simulation; a serial number is appended to the file names.

A *data set info file* (ASCII text format) provides information about the associated *data set file(s)*, i.e. for each associated *data set file* its pathname, the timeframe (start and end) covered and some statistics concerning the interface items logged, i.e. for each item pathname and RID and the number of values contained. The *data set files* contain the logged values in binary format.

Each time a simulator kernel is started, the system generates at least the *log file*. It contains the pathname of the mapping table which maps the pathnames of interface items to their runtime identifiers (RIDs) and the listing of all commands given during the simulation session.

The *log file* and the *archive files* are stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR. The *log file* is marked with the extension .LOG_TXT. The *archive files* are marked with the extension .ARCH_TXT_#, the sharp sign indicates the serial number. In the same directory, the *data set info files* are stored. They are marked with the extension .DATA_SET_INFO_#. The sharp sign indicates the serial number, each file corresponds to the archive file with the same number. For each of these files the part of the name preceeding the extension matches the pattern <model name>_<user name>_<architecture>_DDMMYYYY_HHMM. First comes the simulation model's name (<model name>), then an underline followed by the name of the user (i.e. the name of the user account) who started the simulator kernel (<user name>). Next comes an underline and then the architecture of the host the simulator kernel is/was executing on (<architecture>) followed by another underline. This is followed by 8 characters denoting the date (DDMMYYYY, i.e. day, month and year), a further underline and 4 characters denoting the time (HHMM, i.e. hours and minutes) the simulator kernel was started.

The *data set files* are stored in the UNIX file system in a directory which is determined by the environment variable CSS_LOG_DATA_SET_DIR. Their names match the pattern <model name>_DDMMHHMM_##. First comes the simulation model's name (<model name>), then an underline followed by 4 characters denoting the date (DDMM, i.e. day and month) and 4 characters denoting the time (HHMM, i.e. hours and minutes) the logging results were stored. The serial number (first sharp sign) follows a further underline; it associates the *data set file* to the corresponding *data set info file* with the same number. In order to avoid too large files, a data set may be automatically partitioned furthermore over a subseries of *data set files*; the last character (second sharp sign) denotes the position in this subseries. Because the names of *data set files* are restricted to 20 characters, <model name> may miss some trailing characters of the actual model name.

Copyright per DIN 34

- *If several users are connected to one simulation session (each of them with his/her own logging definitions) there will be only one table with logging results. It is the users task to identify their own logging values.*

- *Warning:*
  Each time you start a model execution the system generates at least the log file (i.e. the file with the extension .LOG_TXT) automatically. It is not possible to suppress this feature. You have to delete this files from time to time or you get problems with your disk quota (i.e. the available space on your computer).

6.4.7.9.5 Restrictions on logging

Logging definitions are subject to the same restrictions as described in section 6.4.7.9.3. It is not possible to log an input connected to the outer border of a composite function block, the same goes for outputs connected to the inner side of a composite.

Interface items of the following types may not be logged to data sets: DURATION, TIME, COMPLEX, all VECTOR types and all MATRIX types.

6.4.7.9.6 Creating tracing items in the simulation table

There are two ways to get the tracing information.

Each time the function block is activated during simulation execution you get the current date, the SMT and the run time ID of the activated FB either into a separate window (**trace to screen**) or written to the log file (**trace to log**). The function blocks marked for tracing are displayed in the simulation table editor window with a grey underlay (see Figure 6–86).

Copyright per DIN 34

Editing tracing elements in a Simulation Table

- Open the Database Browser window.
- Select the model by clicking on its name.
- Press the **Simulation Tables** button.
- Select the Simulation Table by clicking on its name in the list.
- Press and hold the right mouse button. Select **edit** from the pop–up menu. The simulation table editor window appears (in overview mode).
- If the editor window appears in overview mode, move the mouse pointer into the graphic subview and select **inspect** from the pop–up menu.
- Click on the desired function block and select **trace–>to log** from the pop–up menu. The function block is shaded with grey. (see Figure 6–86)

  OR

- Click on the desired function block item and select **trace–>to screen** from the pop–up menu. The function block is shaded with grey.

Note that composite function blocks can‘t be traced.

- Press the **Store Table** button. An input window appears.
- Change the table name in the input field, if the table shall be stored under a new name.
- Press the **Apply** button.
- Choose **Quit** from the window pop–up menu. The table editor window disappears.

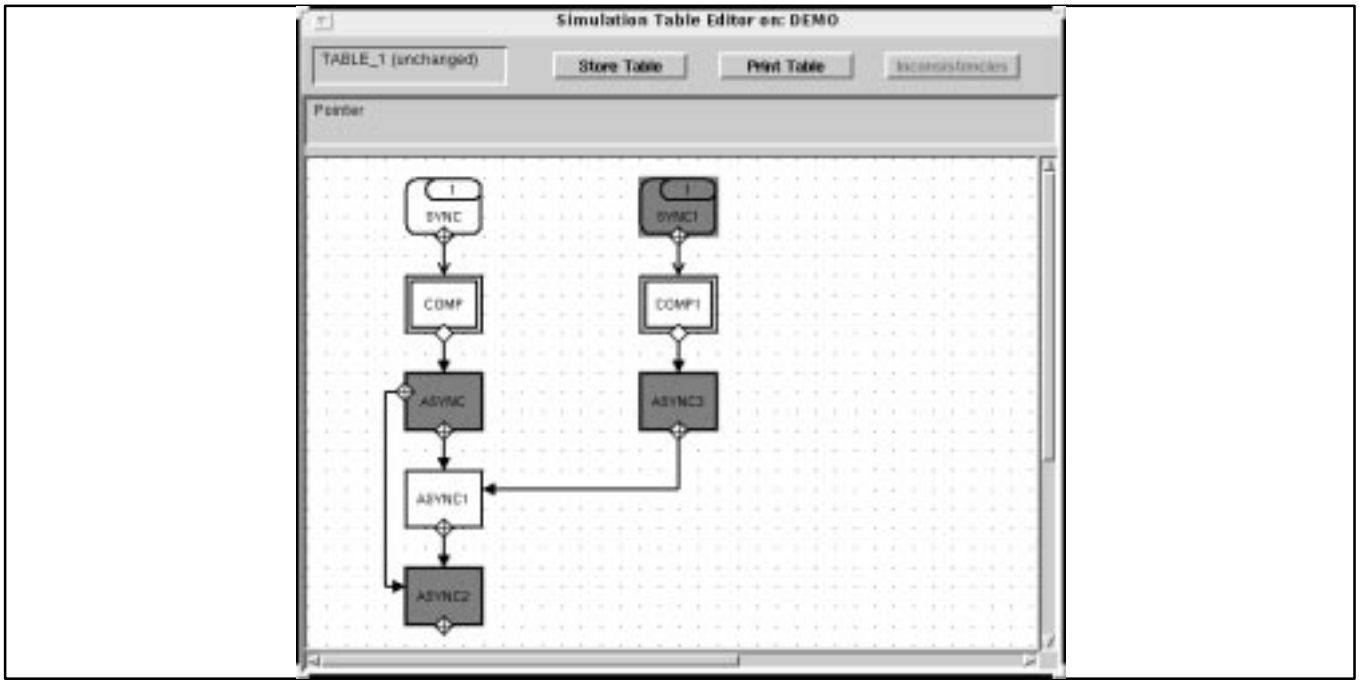


Figure 6–86 : *The function blocks marked for tracing*

Copyright per DIN 34

6.4.7.9.7 Restrictions on tracing

If several users are connected to one simulation session (each of them with his/her own tracing definitions) there will be only one table containing the tracing informations.
The file containing the tracing results will be written as soon as the session owner finishes the simulation session (i.e. the simulator kernel is stopped).

Warning:
If you use the trace to screen option on a synchronous function block running with frame 1 (minframe) and in continuous mode this will block your workstation completely, even a remote login is not possible. The only way to get out of this situation is to re–boot the workstation.
Recommendation : Do NOT use this feature.

Copyright per DIN 34

6.4.8 CSS Configuration Environment Variables

Several shell environment variables control the behaviour of CSS, usually to create additional execution tracing information for troubleshooting. Some may be set in an interactive shell, just before starting the standalone version of CSS (`$CSS_HOME/bin/common/CSS`), some have to be set in the user's `.cshrc` script resp. in a script called `$HOME/.user/css_cshrc` in order to be visible for CSS components started via remote shell or from I_MDB.

- CSS_DEFAULT_STATE_CODE_HANDLING (cshrc)
  set to TRUE to process also the default state code for discrete onboard data types when checking the compatibility to the connected model I/O. Otherwise, the default state code will be ignored.
- CSS_FILE_BASED (interactive)
  set to TRUE to start CSS without connecting to the MDB, working only with models stored in the file system.
- CSS_LOG_DIR (cshrc)
  Directory for storing all CSS logging and debugging output; default is : `$HOME/css/logger`
- CSS_LOG_DATA_SET_DIR (cshrc)
  Directory for data sets produced by the logger_be; default is :
  `$HOME/wd/tev/RESULTS/DATA_SET`
- CSS_MODELS (cshrc)
  Directory for MDB based model files; default is : `$HOME/models.lib`

Variables for synchronization purposes between the CSS kernel and CMAS:

- SET_KERNEL_STATUS_TIMEOUT ($CSS_HOME/user_env/cshrc)
  The value is expressed in milliseconds and defines the time the CSS kernel will wait for an answer from CMAS if the CSS status has changed (e.g. SIMULATION_RUNNING or SIMULATION_STOPPED). The default value is set to 5 seconds, i.e. if the variable is not defined a value of 5000 ms will be assumed. CMAS will need a certain amount of time to initialize internal and associated structures.
- CMAS_INITIALIZATION_TIMEOUT ($CSS_HOME/user_env/cshrc)
  The value is also expressed in milliseconds and defines the allowed time interval for CMAS initialization. The CMAS process is started by the CSS kernel and will initialize internal structures, connected hardware (e.g. MILBus boards) etc. The status of CMAS will be checked by the CSS kernel if a new statevector has to be loaded. In case the CMAS status is not correct and this time interval has expired a warning message will be produced and displayed via the CGS error services. The default value is set to 120 seconds.

Variables for additional debugging information to support troubleshooting:

- CSS_DBS_OPTIONS (cshrc)
  When set, the CSS DB Server will write additional debugging information to files at $CSS_LOG_DIR/db_server.message_log.'hostname'.$$
  Possible settings are : "–v<n>", with n=0..9 indicating different levels of verbosity.
- CSS_LOGGING (cshrc)
  When set to TRUE, the CSS CTG Server will write additional debugging information to files at $CSS_LOG_DIR/ctg_server.message_log.$USER.$HOST.$$

Copyright per DIN 34

6.4.9 File System Maintenance

During preparation and execution of a simulation, CSS saves information in several kinds of log and report files, which eventually ma y consume a significant amount of disk space. Occasionally, it may become necessary to clean up these files, either due to lack of disk space, or just because they are not needed any more.

- **$HOME/CSS_SCOPE_CHECK.LOG** :
  This file accumulates the reports from the CSS scope check window that appears when launched by I_MDB. The latest report will be found at the end, so this file should be frequently deleted to avoid excessive scrolling.
- **$CSS_LOG_DIR** (usually **$HOME/css/logger**) :
  This directory contains log and report files from several CSS components:
  - **<model>_<architecture>_<date>_<time>.LOG_TXT**
    <model>_<architecture>_<date>_<time>.ARCH_TXT_*
    <model>_<architecture>_<date>_<time>.DATA_SET_INFO_* :
    Contain logged data and archived information from simulation execution sessions. This would usually be important simulation results and should be archived somwhere else, or at least be examined carefully, before deletion !
  - **<model>_<architecture>_<date>_<time>.COMPILATION_REPORT_TXT:**
    Contain messages created by the CTG server during compilation of a model. These are just backup copies of the messages displayed in the MDE Configuration Message Window during model configuration and should be deleted occasionally.
  - **kernelMessageOutput*** :
    Contain informational messages by the simulator kernel and CMAS, may be deleted unless needed for troubleshooting
  - **<PID>_<date>_<time>.LOG**
    <PID>_<date>_<time>.ARCH_* :
    Binary 'raw' log and archive files from a (most likely crashed) simulation session. Usually, they are converted to their _TXT counterparts (see above) after normal simulation shutdown by the CSS logger backend program and deleted.
  - **ctg_server.message_log*, db_server.message_log*** :
    Contain informational messages by the ctg– resp. db–server. Creation is usually disabled, but may be activeted for troubleshooting. Files may be deleted unless still needed for troubleshooting.
- **$CSS_SIM_DIR** (usually **$HOME/css/simulations**):
  This directory contains files named **<hostname>@<modelname>@<PID>**, describing currently active simulations, which are deleted during normal shutdown of a simulation. Check occasionally for files left by crashed simulations.
- **$HOME/css/models**:
  This directory contains simulation models that have been exported from the MDB into the file system, distributed over assorted subdirectories. The location of each model is described in the file

Copyright per DIN 34

contents. Exported models that are not needed any more should be deleted using the CSS Database Browser.

- $CSS_MODELS (usually $HOME/models.lib) :
  This directory tree contains all files (source code, Ada libraries, executables and runtime configuration files) related to CSS models that are controlled by the MDB. Paths in this tree are built as :
  <element>–<mission>–<version>/library_sid_<cdu–sid>–V<version>.<issue>.<revision>–<test version>–<instance>/model_sid_<item–sid>

  Note that most information in this directory tree is mandatory for the execution of a CSS simulation model and cannot be re–created once a model's CDU has been frozen in the MDB, so any model's subdirectory in this file system tree should only be deleted when definitely not needed any more (e.g. belonging to an obsolete system tree version).

- Additional model library directories :
  When using the Standalone–File–System version of CSS, arbitrary model library directories may be defined by the user, which are listed in the file **$HOME/.cssdbrc**. Models that are not needed any more should be deleted using the CSS File System Browser.

Copyright per DIN 34

6.4.10 Creating Onboard End Items in the MDB

Note :
This section describes the required settings for configuration of the CGS standard CMAS.
Except for **Software Access Class** *and* **Software Type***, which are required for the creation of an onboard data reference in the model editor, these restrictions may not apply to project specific user defined MDB end items, which are to be handled by project specific CMAS configuration software (cf. section 6.4.7.1.40.3)*

6.4.10.1 End Item Type

The connection of a CSS simulator kernel to an external system (i.e. for a H/W in the loop simulation) is established by creating a number of properly defined onboard end items (stimuli and measurements) in the MDB that then can be referenced from the simulation model's interface items. This information is used by CSS to generate the simulation model specific configuration file allowing the Command and Measurement Adaptation System (CMAS) to initialize itself accordingly.

CGS provides a number of predefined end item types with stimulus resp. measurement characteristic, they can be created via the I_MDB tool. Note that end item types may be restricted to CDUs of specific domains, but by convention almost all end item types may be created in a CDU of domain CGS (e.g. EGSE specific end items may exist only in a CDU of domain EGSE or CGS). Onboard end items may be created directly as subnodes of a CDU; however, if there are many of them, it is usually better to organize them in a tree structure of virtual nodes.

If the end item types provided by CGS are not sufficient, new user defined end item types may be specified with the DADIMA tool prior to creation and installation of the MDB instance. To create a user defined end item type with stimulus or measurement characteristic, set its **Software Access Class** attribute to SEND for a stimulus type resp. READ for a measurement type; CSS will recognize the end item characteristics accordingly.

6.4.10.2 Mapping to CSS Data Type

A model input may reference a stimulus while a model output may reference a measurement. In both cases type compatibility between the referencing interface item and the referenced onboard end item has to be assured, i.e. CSS must be able to map the onboard end item to one of the CSS data types provided at the model interface. This is done by evaluating the item's engineering value type, and, for onboard end items of an integer engineering value type also the item's engineering range.

The engineering value type of a measurement is set when the end item type is specified with the DADIMA tool, it is given by the item's **Software Type** attribute. In contrast, the **Software Type** attribute of a stimulus is always NONE. The engineering value type of a stimulus is set when its formal parameter list (one formal parameter) is compiled with the CLS editor, i.e. it is derived from the UCL type of the formal parameter. An exception are stimuli of the prede-

Copyright per DIN 34

fined end item types PULSE_STIMULUS and BURST_PULSE_STIMULUS: They must not have a formal parameter (consequently the CLS editor cannot be invoked on them), their engineering value type is derived directly from the end item type.

The following table lists all possible engineering value types of onboard end items intended to be referenced by a CSS simulation model together with the corresponding CSS data types provided for simulation model interface items.

Onboard End Item Engineering Value Type	CSS Data Type
INTEGER_TYPE	SIGNED_BYTE SIGNED_SHORT_WORD SIGNED_INTEGER (depending on engineering range)
UNSIGNED_INTEGER_TYPE	UNSIGNED_BYTE UNSIGNED_SHORT_WORD UNSIGNED_INTEGER (depending on engineering range)
REAL_TYPE	REAL
LONG_REAL_TYPE	LONG_REAL
BOOLEAN_TYPE	BOOLEAN
STATE_CODE_TYPE	STATE_CODE
PULSE_TYPE	PULSE
BURST_PULSE_TYPE	BURST_PULSE

The following table lists all possible engineering value types of onboard end items intended to be referenced by a CSS simulation model together with the corresponding UCL types to be specified for the formal parameter of a stimulus end item.

Onboard End Item Engineering Value Type	UCL Type
INTEGER_TYPE	INTEGER
UNSIGNED_INTEGER_TYPE	UNSIGNED_INTEGER
REAL_TYPE	REAL
LONG_REAL_TYPE	LONG_REAL
BOOLEAN_TYPE	BOOLEAN
STATE_CODE_TYPE	STATECODE

For the predefined stimulus end item types, the CLS editor provides a default formal parameter list which should be compiled without any modification. However, for a stimulus of a user defined end item type, the formal parameter list must be specified explicitely as shown in the following example for a stimulus of engineering value type REAL_TYPE (if another engineering value type is desired, replace REAL by the appropriate UCL type):

```
(VALUE: REAL);
```

Copyright per DIN 34

The pulse stimuli are treated specially, their engineering value type is derived directly from the end item type.

Onboard End Item Engineering Value Type	Onboard End Item Type
PULSE_TYPE	PULSE_STIMULUS
BURST_PULSE_TYPE	BURST_PULSE_STIMULUS

A mandatory aggregate for onboard end items of numeric engineering value type is an engineering range, i.e. the **Integer Engineering Range** for onboard end items of engineering value type INTEGER_TYPE, the **Unsigned Integer Engineering Range** for onboard end items of engineering value type UNSIGNED_INTEGER_TYPE resp. BURST_PULSE_TYPE, the **Float Engineering Range** for onboard end items of engineering value type REAL_TYPE resp. the **Double Float Engineering Range** for onboard end items of engineering value type LONG_REAL_TYPE.

In order to map an onboard end item of an integer engineering value type (i.e. engineering value type INTEGER_TYPE resp. UNSIGNED_INTEGER_TYPE) to a CSS data type, CSS evaluates the engineering range defined (i.e. the **Integer Engineering Range** resp. **Unsigned Integer Engineering Range**); the onboard end item is mapped to the smallest possible CSS data type. To avoid confusion, the engineering ranges should be defined as listed in the following table.

Engineering Range **Low Value**..**High Value**	CSS Data Type
–128..+127	SIGNED_BYTE
–32768..+32767	SIGNED_SHORT_WORD
–2147483648..+2147483647	SIGNED_INTEGER
0..255	UNSIGNED_BYTE
0..65535	UNSIGNED_SHORT_WORD
0..2147483647	UNSIGNED_INTEGER

The following table lists all CSS data types available for model interface items together with CGS predefined end item types that may be mapped to them. Note that for onboard end items of integer engineering value type (i.e. UNSIGNED_INTEGER_STIMULUS, UNSIGNED_INTEGER_MEASUREMENT, INTEGER_STIMULUS, EGSE_INTEGER_MEASUREMENT) the end item's engineering range has to be evaluated as described above to allow a non–ambiguous mapping.

CSS Data Type	Predefined Onboard End item Type
UNSIGNED_BYTE	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT
SIGNED_BYTE	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
UNSIGNED_SHORT_WORD	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT

Copyright per DIN 34

SIGNED_SHORT_WORD	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
UNSIGNED_INTEGER	UNSIGNED_INTEGER_STIMULUS UNSIGNED_INTEGER_MEASUREMENT
SIGNED_INTEGER	INTEGER_STIMULUS EGSE_INTEGER_MEASUREMENT
REAL	EGSE_ANALOG_STIMULUS EGSE_FLOAT_MEASUREMENT
LONG_REAL	DOUBLE_FLOAT_STIMULUS DOUBLE_FLOAT_MEASUREMENT
BOOLEAN	BOOLEAN_STIMULUS BOOLEAN_MEASUREMENT
STATE_CODE	EGSE_DISCRETE_STIMULUS EGSE_DISCRETE_MEASUREMENT
PULSE	PULSE_STIMULUS
BURST_PULSE	BURST_PULSE_STIMULUS

The type definition of an onboard end item of engineering value type STATE_CODE must be completed by the specification of the statecode list. The mandatory list of the aggregate **Discrete Calibration** is used for this purpose (this aggregate comprises two attributes, the attribute **Discrete Calibration State Code** is used to specify a state code). Because the list of state codes is part of the type definition, it is relevant for type compatibility with the referencing simulation model's interface item. Both items must refer to the same state codes; however, the order in the respective lists doesn't matter.

The optional attribute **Engineering Units** of onboard end items with engineering value type REAL_TYPE resp. LONG_REAL_TYPE allows to refine the type definition by specifying an engineering unit. An engineering unit is relevant for type compatibility with the referencing simulation model's interface item; both items must refer to the same engineering unit.

6.4.10.3 Physical Address

The following attributes are mandatory (i.e. proper values have to be defined) for simple stimuli and measurements and for CCSDS TM/TC packets. For stimuli describing TC parameters (i.e. parameters contained in a TC packet) and measurements describing TM parameters (i.e. parameters contained in a TM packet), only the attributes **Remote Terminal Slot Class** and **Device Type** are applicable.

Copyright per DIN 34

– **Remote Terminal Slot Class**
::= {BITS_32 | BITS_16 | BITS_8 | BITS_1}
for a simple stimulus resp. measurement
::= PACKET
for a CCSDS TM/TC packet or a stimulus resp. measurement describing a parameter of a CCSDS TM/TC packet
::= NON_STANDARD
for an item transmitted via a Frontend Bus handled by a user defined non–standard driver

Note:
The Remote Terminal Slot Class must be the same for a specific Device Subaddress (Frontend Bus/Device Address/Device Subaddress), i.e. no different slot classes are allowed on one subaddress; this has to be assured for all onboard end items referenced by a specific simulation model.

Note:
On a specific Device Subaddress (Frontend Bus/Device Address/Device Subaddress), there may exist either only stimuli or only measurements; this has to be assured for all onboard end items referenced by a specific simulation model.

Note:
A complete Frontend Bus may be handled by a user defined non–standard driver linked with standard CMAS (e.g. it is not possible to declare a specific Device Subaddress as NON_STANDARD); the Frontend Bus must be configured to support NON_STANDARD (i.e. only NON_STANDARD) in the CMAS VME config file.

Note:
The onboard end item related information as specified in the MDB (i.e. physical address, packet definition, (de)calibration definition) is available also for a non–standard driver; however, it may or may not make use of this information, as appropriate.

– **Device Type**
::= MIL_BUS

– **Frontend Bus**
::= 0..2147483647
logical bus (index) referred to in the CMAS VME config file

Note:
For each simulation model, the Frontend Busses handled must be indexed strictly sequential starting with 0 (i.e. 0, 1, 2, 3, ..., n–1 with n giving the number of Frontend Busses handled)

– **Device Address**
::= {0..30}
MIL1553–Bus RT address (address 31 is reserved for broadcasting)

– **Device Subaddress**
::= {1..30}
MIL1553–Bus RT subaddress (addresses 0 and 31 are reserved for mode codes allowing to command the RT itself)

Copyright per DIN 34

– **Device Channel**
::= {0..15} for **Remote Terminal Slot Class** BITS_32
::= {0..31} for **Remote Terminal Slot Class** BITS_16
::= {0..63} for **Remote Terminal Slot Class** BITS_8
::= {0..511} for **Remote Terminal Slot Class** BITS_1 or NON_STANDARD
::= {0..0} for **Remote Terminal Slot Class** PACKET
512 bits (i.e. 32 16–bit data words) is the maximum size of a single MIL–Bus message

The following attributes are optional for onboard end items of **Remote Terminal Slot Class** NON_STANDARD; they may be used to provide information for the control of a user defined non–standard driver (standard CMAS does not evaluate them).

– **Command 1**
::= STRING
some arbitrary string

– **Command 2**
::= STRING
some arbitrary string

6.4.10.4 CCSDS TM/TC Packets

A mandatory attribute of a stimulus describing a TC parameter or a measurement describing a TM parameter is the **Raw Value Size in Bits** (this attribute is not applicable for simple stimuli and measurements since the information is implicitely defined by their **Remote Terminal Slot Class** attribute). Such stimuli and measurements must be referenced from the packets they are contained in.

A TM packet (from CSS point of view a packet that is received by the external system from the simulator kernel) is described by an end item of type CCSDS_ADU_DESCRIPTION. Mandatory aggregates of this end item are the **CCSDS Primary Header** and the **Physical Address** (see above), an optional attribute is the **CCSDS Second Header**. The list of the aggregate **Measurement End Items** must contain an entry for each TM parameter. This aggregate comprises the attributes **End Item Reference**, i.e. a reference to a measurement end item, and **Location**, i.e. the bit offset in the CCSDS User Data Field (1..2147483647).

A TC packet (from CSS point of view a packet that is sent from the external system to the simulator kernel) is described by an end item of type EGSE_PREDEFINED_TC. Mandatory aggregates of this end item are the **CCSDS Primary Header** and the **Physical Address** (see above), an optional attribute is the **CCSDS Second Header**. The list of the aggregate **TC End Item References** must contain an entry for each TC parameter. This aggregate comprises the attributes **Stimulus Reference**, i.e. a reference to a stimulus end item, and **Location**, i.e. the bit offset in the CCSDS User Data Field (1..32768).

Note: Inputs of CSS simulation models can only be connected to (i.e. reference) stimulus end items. Originally the TC parameters of an EGSE_PREDEFINED_TC are specified by the CLS Editor together with a corresponding list of the aggregate **List of Parameters** (the com-

Copyright per DIN 34

mand's parameters) and a list of the aggregate **General Bitstream Layout** together with the corresponding lists of definition aggregates, i.e. **Integer Definition**, **Float Definition** and **Binary Definition** (the command's constant patterns). CSS will see neither the parameters nor the constant patterns. The **TC End Item References** list was introduced especially (and exclusively) for CSS. It allows to emulate the EGSE_PREDEFINED_TC's parameters and constant patterns redundantly with a number of separate stimulus end items. This list must comply exactly in engineering value types and locations (i.e. bit offsets) with the EGSE_PREDEFINED_TC's parameters and constant patterns as specified by the aggregates listed above.

6.4.10.5 Calibration/Decalibaration Definition

The meaning of calibration and decalibration is as follows:

Calibration
transformation of a raw value into an engineering value

Decalibration
transformation of an engineering value into a raw value

From the Ground System's point of view, stimuli have to be decalibrated while measurements have to be calibrated (CMAS performs the inverse operation for CSS: decalibration of measurements, calibration of stimuli).

Numeric onboard end items (i.e. end items of engineering value type INTEGER_TYPE, UNSIGNED_INTEGER_TYPE, REAL_TYPE, LONG_REAL_TYPE or BURST_PULSE_TYPE)

The following attributes are mandatory (i.e. proper values have to be defined):

– **Raw Value Type**
::= SIGNED_INTEGER for onboard end items of engineering value type INTEGER_TYPE
::= UNSIGNED_INTEGER for onboard end items of engineering value type
UNSIGNED_INTEGER_TYPE or BURST_PULSE_TYPE
::= {UNSIGNED_INTEGER | SIGNED_INTEGER} for onboard end items of engineering value
type REAL_TYPE or LONG_REAL_TYPE

– **Calib Curve Type**
::= {POINT_PAIRS | POLYNOM | IDENTICAL}

A mandatory aggregate is the raw value range, i.e. the **Integer Raw Value Range** for an item with **Raw Value Type** SIGNED_INTEGER resp. the **Unsigned Integer Raw Value Range** for an item with **Raw Value Type** UNSIGNED_INTEGER.

Another mandatory aggregate is the (de)calibration definition for end items with **Calib Curve Type** POINT_PAIRS or POLYNOM, i.e. **Analog Decalibration Point Pairs** for a stimulus end item with **Calib Curve Type** POINT_PAIRS, **Analog Point Pairs** for a measurement end item with **Calib Curve Type** POINT_PAIRS, **Analog Decalibration Coeffi-**

Copyright per DIN 34

cients for a stimulus end item with **Calib Curve Type** POLYNOM resp. **Analog Calibration Coefficients** for a measurement end item with **Calib Curve Type** POLYNOM.

Note:
The (de)calibaration definition must allow inversion, i.e. each raw value must be associated to exactly one engineering value and vice versa (unique relation raw value–engineering value).

Note:
The (de)calibaration definition must assure that the range limits computed (raw value range limits resp. engineering range limits) match exactly the respective range limit specification, which may be tricky for polynom (de)calibration because of possible rounding errors.

An optional aggregate for end items with **Calib Curve Type** POLYNOM is the (de)calibration point pairs allowing to determine the conversion in inverse direction, i.e. **Analog Point Pairs** for a stimulus end item resp. **Analog Decalibration Point Pairs** for a measurement end item. If not explicitely specified, point pairs for the conversion in inverse direction will be calculated automatically from the polynom.

Discrete onboard end items (i.e. end items of engineering value type STATE_CODE_TYPE)

The following attribute and aggregate are mandatory:

- **Raw Value Type**
  ::= {UNSIGNED_INTEGER | INTEGER}
- **Discrete Calibration**
  a list of the aggregate **Discrete Calibration**; this aggregate comprises the attributes **Discrete Calibration State Code** and **Discrete Calibration Raw Value**, i.e. a state code specification and the associated raw value

Another mandatory aggregate is the raw value range, i.e. the **Integer Raw Value Range** for an item with **Raw Value Type** SIGNED_INTEGER resp. the **Unsigned Integer Raw Value Range** for an item with **Raw Value Type** UNSIGNED_INTEGER.

Boolean onboard end items (i.e. end items of engineering value type BOOLEAN_TYPE or PULSE_TYPE)

The following attribute and aggregate are mandatory:

- **Raw Value Type**
  ::= {UNSIGNED_INTEGER | INTEGER}
- **Boolean Calibration**
  this aggregate comprises the attributes **True** and **False** allowing to specify the respective raw values

6.4.10.6 Feedback

When an onboard end item is selected in the CSS Database Browser, CSS provides its characteristic (stimulus resp. measurement), the CSS data type it is mapped to (NONE if mapping is not possible), and, if the definition of an onboard end item is erroneous (e.g. there are mandatory attributes missing), also an error message (e.g. no raw value type, no raw value range, no calibration description, no coefficients, no physical address, etc.).

Copyright per DIN 34

6.4.11 Description of mathematical constants and routines delivered with CSS

—
— KEYWORDS — CSS_MATH
— Project CSS
— ADD, section 5.3 (CSS Repository)
— HOOD Object CSS_MATH
—

This description shall support the user's model SW development. The "undermentioned" mathematical constants and routines may be used within the AIL (Atomic Implementation Language, i.e. Ada subset) of a synchronous or asynchronous Function Block and are annotated as far as necessary.

— **CONSTANTS**

Values have been taken from the following sources:

- CRC Handbook of Tables for Mathematics, Fourth Ed., Robert C. Weast (ed.), 1970, The Chemical Rubber Co.
- Knuth, Donald E., 'Fundamental Algorithms', Vol. 1 of 'The Art of Computer Programming', 2nd ed., 1973. (Appendix B).
- Davis, Harold T. and Fisher, Vera J., 'Arithmetical Tables', Vol. III of 'Tables of the Mathematical Functions', The Principia Press, Texas, 1962.
- Fletcher, A. et al., 'An Index of Mathematical Tables', Scientific Computing Service Limited, London, 1962.

Where values exist in more than one source, such values have been cross checked. In all cases, such values agree except for possibly a value of one in the last digit. In such cases of difference, the higher value is used, under the assumption that it is a rounded value and that the lower value is a truncated value.

The list of constants specified is a combination of lists in the various sources and the list in the Elementary Function Package Proposal. Multiples and inverses of constants, which are computable at compile time with full accuracy, have been eliminated. The only exceptions are 1.0/pi and 1.0/e which are so frequently encountered that their definition is virtually "forced."

Naming is a combination of suggested names in the Elementary Function Package Proposal, non–conflict with likely variable names (single letter disallowed), descriptive naming (for in–use recognition), and a

Copyright per DIN 34

length decision. Names are shorter for constants that are more frequently used and more likely to be known. The naming criteria sometimes conflict with the desire for naming consistency.

Hence we have BASE_E (no single letter names) but E is used in all combination names. Similarly, GOLDEN_RATIO is spelled out (PHI likely to conflict with user variable names), but PHI is used for all composite names. The hardest decision was choosing BIN_LOG.., NAT_LOG.., and COM_LOG.. over the often used LN2, LN, and LOG.

Copyright per DIN 34

```
------------------------------------------------------------------------------
------------------------------------------------------------------------------


--  Basic constants :
--
--   pi, e, 1/pi, 1/e :
--
PI        : constant :=  3.14159_26535_89793_23846_26433_83279_50288;
BASE_E    : constant :=  2.71828_18284_59045_23536_02874_71352_66250;
INV_PI    : constant :=  0.31830_98861_83790_67153_77675_26745_02872;
INV_E     : constant :=  0.36787_94411_71442_32159_55237_70161_46087;
--
--   Square roots :
--
SQRT_2    : constant :=  1.41421_35623_73095_04880_16887_24209_69808;
SQRT_3    : constant :=  1.73205_08075_68877_29352_74463_41505_87237;
SQRT_5    : constant :=  2.23606_79774_99789_69640_91736_68731_27624;
SQRT_7    : constant :=  2.64575_13110_64590_59050_16157_53639_26043;
SQRT_10   : constant :=  3.16227_76601_68379_33199_88935_44432_71853;
SQRT_PI   : constant :=  1.77245_38509_05516_02729_81674_83341_14518;
--
--   Cube and fourth roots :
--
CBRT_2    : constant :=  1.25992_10498_94873_16476_72106_07278_22835;
CBRT_3    : constant :=  1.44224_95703_07408_38232_16383_10780_10959;
CBRT_4    : constant :=  1.58740_10519_68199_47475;
CBRT_10   : constant :=  2.15443_46900_31883_72715;
FOURTH_RT_2    : constant :=  1.18920_71150_02721_06671_74999_70560_47592;
FOURTH_RT_10   : constant :=  1.77827_94100_38922_80122;
--
--
--   Common expressions with pi and e :
--
--  pi**2, pi**e, e**2, e**e, e**(1/e), e**(-e), e**(-1/e),
--  e**pi, e**(pi/2), e**(pi/4); e**(-pi), e**(-pi/2), e**(-p/4)
--  Note: i**i = e**(-pi/2) where i is the unit imaginary value
--
PI_SQ     : constant :=  9.86960_44010_89358_61883_44909_99876_15114;
PI_TO_E   : constant := 22.45915_77183_61045_47342_71522_045;
EXP_SQ    : constant :=  7.38905_60989_30650_22723_04274_60575_00781;
EXP_E     : constant := 15.15426_22414_79264;
EXP_INV_E : constant :=  1.44466_78610_09766;
EXP_NEG_E : constant :=  0.06598_80358_45312;
EXP_NEG_INV_E  : constant :=  0.69220_06275_55346;
EXP_PI    : constant := 23.14069_26327_79269_00572_90864;
EXP_HALF_PI    : constant :=  4.81047_73809_65351_65547_30357;
EXP_4TH_PI     : constant :=  2.19328_00507_38015_45655_97696_59278_73822;
EXP_NEG_PI     : constant :=  0.04321_39182_63772_24977_44177;
EXP_NEG_HALF_PI    : constant :=  0.20787_95763_50761_90854_69556;
EXP_NEG_4TH_PI: constant :=  0.45593_81277_65996_23676_59212;
--
```

Copyright per DIN 34

```
--   Base e logarithms :
--
NAT_LOG_2     : constant :=  0.69314_71805_59945_30941_72321_21458_17657;
NAT_LOG_3     : constant :=  1.09861_22886_68109_69139_52452_36922_52570;
NAT_LOG_10    : constant :=  2.30258_50929_94045_68401_79914_54684_36421;
NAT_LOG_E     : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
NAT_LOG_PI    : constant :=  1.14472_98858_49400_17414_34273_51353_05871;
--
--   Base 10 logarithms :
--
COM_LOG_2     : constant :=  0.30102_99956_63981_19521_37388_94724_49303;
COM_LOG_3     : constant :=  0.47712_12547_19662_43729_50279_03255_11531;
COM_LOG_10    : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
COM_LOG_E     : constant :=  0.43429_44819_03251_82765_11289_18916_60508;
COM_LOG_PI    : constant :=  0.49714_98726_94133_85435_12682_88290_89887;
--
--   Base 2 logarithms :
--
BIN_LOG_2     : constant :=  1.00000_00000_00000_00000_00000_00000_00000;
BIN_LOG_3     : constant :=  NAT_LOG_3  / NAT_LOG_2 ;
BIN_LOG_10    : constant :=  NAT_LOG_10 / NAT_LOG_2 ;
BIN_LOG_E     : constant :=  1.0        / NAT_LOG_2 ;
BIN_LOG_PI    : constant :=  NAT_LOG_PI / NAT_LOG_2 ;
--
--   Golden ratio, phi ( a/b where a/b = b/(a-b) ) :
--
GOLDEN_RATIO  : constant :=  1.61803_39887_49894_84820_45868_34365_63812;
NAT_LOG_PHI   : constant :=  0.48121_18250_59603_44749_77589_13424_36842;
--
--   Euler's constant (gamma), log(base e) gamma, e**gamma, e**(-gamma) :
--
GAMMA         : constant :=  0.57721_56649_01532_86060_65120_90082_40243;
NAT_LOG_GAMMA : constant := -0.54953_93129_81644_82233_7662 ;
COM_LOG_GAMMA : constant := -1.76133_81087_83167_61054_0;
EXP_GAMMA     : constant :=  1.78107_24179_90197_98523_65041_03107_17955;
EXP_NEG_GAMMA : constant :=  0.56145_94835_66885_16983;
--
--   Trigonometric and hyperbolic values for 1.0 radian :
--
SIN_1         : constant :=  0.84147_09848_07896_50665_25023_21630_29900;
COS_1         : constant :=  0.54030_23058_68139_71740_09366_07442_97660;
TAN_1         : constant :=  1.55740_77246_54902;
SINH_1        : constant :=  1.17520_11936_43801;
COSH_1        : constant :=  1.54308_06348_15244;
TANH_1        : constant :=  0.76159_41559_55764;
--
```

Copyright per DIN 34

```
------------------------------------------------------------------------
--  EXCEPTIONS
------------------------------------------------------------------------
--

DOMAIN_ERROR  : raised if UNIX errno returns EDOM, indicating the arguments were
not valid for the function

RANGE_ERROR   : raised if UNIX errno returns ERANGE, indicating that the correct
value cannot be computed


--
------------------------------------------------------------------------
--  FUNCTION SPECIFICATIONS
------------------------------------------------------------------------
--

-- Exponential, logarithm and root routines :
--
-- Long_Float versions :
--
function EXP (X : LONG_REAL) return LONG_REAL;      -- e**x
    -- raises RANGE_ERROR if correct value would overflow

function NAT_LOG (X : LONG_REAL) return LONG_REAL;  -- log base e (x)
function LOG(X : LONG_REAL) return LONG_REAL renames NAT_LOG;
    -- raises DOMAIN_ERROR if x is zero or negative

function COM_LOG (X : LONG_REAL) return LONG_REAL;  -- log base 10 (x)

function BIN_LOG (X : LONG_REAL) return LONG_REAL;  -- log base 2 (x)
    -- raises DOMAIN_ERROR if x is zero or negative

function "**" (X,Y : LONG_REAL) return LONG_REAL;   -- x**y
    -- raises RANGE_ERROR if correct value would overflow
    -- raises DOMAIN_ERROR when first argument is negative and second is
    -- non-integer or when first argument is 0 and second is <= 0

function SQRT (X : LONG_REAL) return LONG_REAL;     -- x**(1/2)
    -- raises DOMAIN_ERROR when x is negative

--
--  Short_float versions :
--

function EXP (X : REAL) return REAL;
    -- raises RANGE_ERROR if correct value would overflow

function NAT_LOG (X : REAL) return REAL;
function LOG(X : REAL) return REAL renames NAT_LOG;
    -- raises DOMAIN_ERROR if x is zero or negative
```

Copyright per DIN 34

```
function COM_LOG (X : REAL) return REAL;

function BIN_LOG (X : REAL) return REAL;
    -- raises DOMAIN_ERROR if x is zero or negative

function "**" (X,Y : REAL) return REAL;
    -- raises RANGE_ERROR if correct value would overflow
    -- raises DOMAIN_ERROR when first argument is negative and second is
    -- non-integer or when first argument is 0 and second is <= 0

function SQRT (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative


--
--  Floor and ceiling functions :

--
-- Long_Float versions :
--
function FLOOR (X : LONG_REAL) return LONG_REAL;
function CEIL (X : LONG_REAL) return LONG_REAL;

--
-- Short_float versions :

--
function FLOOR (X : REAL) return REAL;
function CEIL (X : REAL) return REAL;



--
--  Gamma functions :

--
type SIGN_TYPE is (NEGATIVE, POSITIVE);
--
-- Long_Float versions :
--
type LONG_REAL_GAMMA_COMPONENTS is record
    LOG_OF_GAMMA : LONG_REAL;
    SIGN : SIGN_TYPE;
    end record;


function CALC_GAMMA(X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer

function LOG_GAMMA(X : LONG_REAL) return LONG_REAL_GAMMA_COMPONENTS;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer

--
-- Short_float versions :
```

Copyright per DIN 34

```
--
type REAL_GAMMA_COMPONENTS is record
    LOG_OF_GAMMA : REAL;
    SIGN : SIGN_TYPE;
    end record;

function CALC_GAMMA (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer
function LOG_GAMMA(X : REAL) return REAL_GAMMA_COMPONENTS;
    -- raises DOMAIN_ERROR when x <= 0 and x is an integer



--
--  Rectangular to polar coordinates functions :
--
```

Note:
ANGLE *returns radians result, range –pi/2 .. +pi/2.* ***ANGLE*** *is ARCTAN (1st_arg/2nd_arg). The call* ***ANGLE (yc,xc)*** *(where yc is the y–axis component and xc is the x–axis component) yields the angle from the x–axis to the complex value specified by (xc,yc). The call* ***ANGLE (x=>xc,y=>yc)*** *is identical to the above call.*

```
--
-- Long_Float versions :
--
function RADIUS (X,Y : LONG_REAL) return LONG_REAL;

function ANGLE (Y,X : LONG_REAL) return LONG_REAL;

--
-- Short_float versions :
--

function RADIUS (X,Y : REAL) return REAL;
function ANGLE (Y,X : REAL) return REAL;



--
--  Bessel functions :--
--
-- Long_Float versions :
--
function J0 (X : LONG_REAL) return LONG_REAL;
function J1 (X : LONG_REAL) return LONG_REAL;
function JN (N : INTEGER; X : LONG_REAL) return LONG_REAL;

function Y0 (X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative

function Y1 (X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative

function YN (N : INTEGER; X : LONG_REAL) return LONG_REAL;
    -- raises DOMAIN_ERROR when x is negative
```

Copyright per DIN 34

```
--
-- Short_float versions :
--

function J0 (X : REAL) return REAL;
function J1 (X : REAL) return REAL;
function JN (N : INTEGER; X : REAL) return REAL;

function Y0 (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative

function Y1 (X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative

function YN (N : INTEGER; X : REAL) return REAL;
    -- raises DOMAIN_ERROR when x is negative


--
--    Basic trigonometric functions :
--
--  NOTE:  Inputs are radians, outputs are unit-less ratios

--
-- Long_Float versions :
--

function SIN (X : LONG_REAL) return LONG_REAL;
function COS (X : LONG_REAL) return LONG_REAL;

function TAN (X : LONG_REAL) return LONG_REAL;
    -- raises RANGE_ERROR for singular points
    -- value of TAN is garbage for x > 2**31

--
-- Short_float versions :
--

function SIN (X : REAL) return REAL;

function COS (X : REAL) return REAL;

function TAN (X : REAL) return REAL;
    -- raises RANGE_ERROR for singular points
    -- value of TAN is garbage for x > 2**31


--
--    Basic inverse trigonometric functions :
--
--  NOTE: Inputs are unit-less ratios, outputs are radians

--
-- Long_Float versions :
--
```

Copyright per DIN 34

```
function ARCSIN (X : LONG_REAL) return LONG_REAL;
    -- –1 <= x ,= +1
    -- range is –pi/2 .. +pi/2
    -- raises DOMAIN_ERROR when x > 1
function ARCCOS (X : LONG_REAL) return LONG_REAL;
    -- –1 <= x ,= +1
    -- range is 0 .. +pi
    -- raises DOMAIN_ERROR when x > 1

function ARCTAN (X : LONG_REAL) return LONG_REAL;
function ATAN (X : LONG_REAL) return LONG_REAL renames ARCTAN;
    -- x is unbounded
    -- range is –pi/2 .. +pi/2

--
-- Short_float versions :
--

function ARCSIN (X : REAL) return REAL;
    -- –1 <= x ,= +1
    -- range is –pi/2 .. +pi/2
    -- raises DOMAIN_ERROR when x > 1

function ARCCOS (X : REAL) return REAL;
    -- –1 <= x ,= +1
    -- range is 0 .. +pi
    -- raises DOMAIN_ERROR when x > 1

function ARCTAN (X : REAL) return REAL;
function ATAN (X : REAL) return REAL renames ARCTAN;
    -- x is unbounded
    -- range is –pi/2 .. +pi/2



--
--    Basic inverse trigonometric functions :

--
-- Long_Float versions :
--

function SINH (X : LONG_REAL) return LONG_REAL;
    -- returns huge value with appropriate sign when correct value would overflow

function COSH (X : LONG_REAL) return LONG_REAL;
    -- returns huge value with appropriate sign when correct value would overflow

function TANH (X : LONG_REAL) return LONG_REAL;

--
-- Short_float versions :
--

function SINH (X : REAL) return REAL;
    -- returns huge value with appropriate sign when correct value would overflow
```

Copyright per DIN 34

```
function COSH (X : REAL) return REAL;
    -- returns huge value with appropriate sign when correct value would overflow

function TANH (X : REAL) return REAL;


------------------------------------------------------------------------------
------------------------------------------------------------------------------

--
--
-- DISTRIBUTION AND COPYRIGHT :
--
This software is copyright 1995 by DASA-RI, RIT14, Bremen, Germany. All rights re-
served.  No part of the material protected by this copyright notice may be repro-
duced or utilized in any form without written permission of the copyright owner.
--
--
```

6.4.12 CSS Data Types

```
------------------------------------------------------------------------------
--
-- ABSTRACT -- CSS_TYPES
-- Provides the set of types defined for CSS MDE model variables.

------------------------------------------------------------------------------
--
-- KEYWORDS -- CSS_TYPES
-- Project CSS-Downsizing
-- <document refs; e.g.: DDD section 3.2>
-- HOOD object CSS_TYPES
-- MDE model variable types

------------------------------------------------------------------------------
--
-- CONTENTS -- CSS_TYPES
--
-- Type:      Package Spec (+ implicit body)
--
-- Version:
-- 2.1, 10.11.93, J.Hoyng: initial version
-- 3.1, 21.07.94, J.Hoyng: the TIME type depends now on SMT_CALENDAR.TIME
--                  deleted type declaration DURATION
-- 3.2, 06.04.95, J.Hoyng: implemented new types
-- 4.1, 23.08.95, J.Hoyng: build_2 version
--
-- Purpose:
--   Provide CSS defined typeset
--
-- External:
-- From Ada system:
--   CALENDAR
--
-- From CGS:
--   NUMERIC_TYPES
--
-- From CSS:
--   DB_DEFINITIONS
--      SMT_CALENDAR
--
-- Machine Dependencies:
--   none
--
-- Compiler Dependencies:
--   none
------------------------------------------------------------------------------

£
-- imports from Ada system:
with CALENDAR;
```

Copyright per DIN 34

```
-- imports from CGS:
with NUMERIC_TYPES;

-- imports from CSS:
with DB_DEFINITIONS;
with SMT_CALENDAR;

package CSS_TYPES is

type CSS_SW_TYPES is (
         NONE,
         UNSIGNED_BYTE_TYPE,
         SIGNED_BYTE_TYPE,
         UNSIGNED_SHORT_WORD_TYPE,
         SIGNED_SHORT_WORD_TYPE,
         UNSIGNED_INTEGER_TYPE,
         SIGNED_INTEGER_TYPE,
         REAL_TYPE,
         LONG_REAL_TYPE,
         COMPLEX_TYPE,
         BOOLEAN_TYPE,
         TIME_TYPE,
         LONG_DURATION_TYPE,
         STATE_CODE_TYPE,
         PULSE_TYPE,
         BURST_PULSE_TYPE,
         UNSIGNED_BYTE_VECTOR_TYPE,
         SIGNED_BYTE_VECTOR_TYPE,
         UNSIGNED_SHORT_WORD_VECTOR_TYPE,
         SIGNED_SHORT_WORD_VECTOR_TYPE,
         UNSIGNED_INTEGER_VECTOR_TYPE,
         SIGNED_INTEGER_VECTOR_TYPE,
         REAL_VECTOR_TYPE,
         LONG_REAL_VECTOR_TYPE,
         COMPLEX_VECTOR_TYPE,
         BOOLEAN_VECTOR_TYPE,
         UNSIGNED_BYTE_MATRIX_TYPE,
         SIGNED_BYTE_MATRIX_TYPE,
         UNSIGNED_SHORT_WORD_MATRIX_TYPE,
         SIGNED_SHORT_WORD_MATRIX_TYPE,
         UNSIGNED_INTEGER_MATRIX_TYPE,
         SIGNED_INTEGER_MATRIX_TYPE,
         REAL_MATRIX_TYPE,
         LONG_REAL_MATRIX_TYPE,
         COMPLEX_MATRIX_TYPE,
         BOOLEAN_MATRIX_TYPE,
         BYTE_STREAM_TYPE,
         LAST_TYPE);

-- all CSS_SW_TYPES used during runtime
subtype CSS_RUNTIME_SW_TYPES is CSS_SW_TYPES range
      CSS_SW_TYPES'SUCC(NONE) .. CSS_SW_TYPES'PRED (LAST_TYPE);
```

Copyright per DIN 34

```
type UNSIGNED_INTEGER is new NUMERIC_TYPES.UNSIGNED_INTEGER32
                     range 0 .. NUMERIC_TYPES.UNSIGNED_INTEGER32'LAST;

------------------------------
-- SIGNED_BYTE
------------------------------
type SIGNED_BYTE is new NUMERIC_TYPES.INTEGER8;

SIGNED_BYTE_INITIAL_VALUE : constant SIGNED_BYTE := 0;

type SIGNED_BYTE_VECTOR is
       array (UNSIGNED_INTEGER range <>) of SIGNED_BYTE;

type SIGNED_BYTE_VECTOR_ACCESS is access SIGNED_BYTE_VECTOR;

type SIGNED_BYTE_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_BYTE;

type SIGNED_BYTE_MATRIX_ACCESS is access SIGNED_BYTE_MATRIX;

------------------------------
--  UNSIGNED_BYTE
------------------------------
type UNSIGNED_BYTE is new NUMERIC_TYPES.BYTE;

UNSIGNED_BYTE_INITIAL_VALUE : constant UNSIGNED_BYTE := 0;

type UNSIGNED_BYTE_VECTOR is array (UNSIGNED_INTEGER range <>) of UNSIGNED_BYTE;

type UNSIGNED_BYTE_VECTOR_ACCESS is access UNSIGNED_BYTE_VECTOR;

type UNSIGNED_BYTE_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_BYTE;

type UNSIGNED_BYTE_MATRIX_ACCESS is access UNSIGNED_BYTE_MATRIX;

------------------------------
--  UNSIGNED_SHORT_WORD
------------------------------
type UNSIGNED_SHORT_WORD is new NUMERIC_TYPES.UNSIGNED_INTEGER16;

UNSIGNED_SHORT_WORD_INITIAL_VALUE : constant UNSIGNED_SHORT_WORD := 0;

type UNSIGNED_SHORT_WORD_VECTOR is
       array (UNSIGNED_INTEGER range <>) of UNSIGNED_SHORT_WORD;

type UNSIGNED_SHORT_WORD_VECTOR_ACCESS is access UNSIGNED_SHORT_WORD_VECTOR;

type UNSIGNED_SHORT_WORD_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_SHORT_WORD;

type UNSIGNED_SHORT_WORD_MATRIX_ACCESS is access UNSIGNED_SHORT_WORD_MATRIX;

------------------------------
-- SIGNED_SHORT_WORD
------------------------------
type SIGNED_SHORT_WORD is new NUMERIC_TYPES.INTEGER16;
```

Copyright per DIN 34

```
SIGNED_SHORT_WORD_INITIAL_VALUE : constant SIGNED_SHORT_WORD := 0;

type SIGNED_SHORT_WORD_VECTOR is
        array (UNSIGNED_INTEGER range <>) of SIGNED_SHORT_WORD;

type SIGNED_SHORT_WORD_VECTOR_ACCESS is access SIGNED_SHORT_WORD_VECTOR;

type SIGNED_SHORT_WORD_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_SHORT_WORD;

type SIGNED_SHORT_WORD_MATRIX_ACCESS is access SIGNED_SHORT_WORD_MATRIX;

------------------------------
-- SIGNED_INTEGER
------------------------------
type SIGNED_INTEGER is new NUMERIC_TYPES.INTEGER32;

SIGNED_INTEGER_INITIAL_VALUE : constant SIGNED_INTEGER := 0;

type SIGNED_INTEGER_VECTOR is
       array (UNSIGNED_INTEGER range <>) of SIGNED_INTEGER;

type SIGNED_INTEGER_VECTOR_ACCESS is access SIGNED_INTEGER_VECTOR;

type SIGNED_INTEGER_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of SIGNED_INTEGER;

type SIGNED_INTEGER_MATRIX_ACCESS is access SIGNED_INTEGER_MATRIX;

------------------------------
--  UNSIGNED_INTEGER
------------------------------

UNSIGNED_INTEGER_INITIAL_VALUE : constant UNSIGNED_INTEGER := 0;

type UNSIGNED_INTEGER_VECTOR is
       array (UNSIGNED_INTEGER range <>) of UNSIGNED_INTEGER;

type UNSIGNED_INTEGER_VECTOR_ACCESS is access UNSIGNED_INTEGER_VECTOR;

type UNSIGNED_INTEGER_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of UNSIGNED_INTEGER;

type UNSIGNED_INTEGER_MATRIX_ACCESS is access UNSIGNED_INTEGER_MATRIX;

------------------------------
-- REAL
------------------------------
type REAL is new NUMERIC_TYPES.SINGLE_FLOAT;

REAL_INITIAL_VALUE : constant REAL := 0.0;

type REAL_VECTOR is
       array (UNSIGNED_INTEGER range <>) of REAL;

type REAL_VECTOR_ACCESS is access REAL_VECTOR;
```

Copyright per DIN 34

```
type REAL_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of REAL;

type REAL_MATRIX_ACCESS is access REAL_MATRIX;

------------------------------
-- LONG_REAL
------------------------------
type LONG_REAL is new NUMERIC_TYPES.DOUBLE_FLOAT;

LONG_REAL_INITIAL_VALUE : constant LONG_REAL := 0.0;

type LONG_REAL_VECTOR is
       array (UNSIGNED_INTEGER range <>) of LONG_REAL;

type LONG_REAL_VECTOR_ACCESS is access LONG_REAL_VECTOR;

type LONG_REAL_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of LONG_REAL;

type LONG_REAL_MATRIX_ACCESS is access LONG_REAL_MATRIX;

------------------------------
-- COMPLEX
------------------------------

type COMPLEX is
  record
    RE, IM :  LONG_REAL;
  end record;

COMPLEX_INITIAL_VALUE : constant COMPLEX := (0.0, 0.0);

type COMPLEX_VECTOR is
       array (UNSIGNED_INTEGER range <>) of COMPLEX;

type COMPLEX_VECTOR_ACCESS is access COMPLEX_VECTOR;

type COMPLEX_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of COMPLEX;

type COMPLEX_MATRIX_ACCESS is access COMPLEX_MATRIX;

------------------------------
-- BOOLEAN
------------------------------
BOOLEAN_INITIAL_VALUE : constant BOOLEAN := TRUE;

type BOOLEAN_VECTOR is
       array (UNSIGNED_INTEGER range <>) of BOOLEAN;

type BOOLEAN_VECTOR_ACCESS is access BOOLEAN_VECTOR;

type BOOLEAN_MATRIX is
       array (UNSIGNED_INTEGER range <>,
              UNSIGNED_INTEGER range <>) of BOOLEAN;
```

Copyright per DIN 34

```
type BOOLEAN_MATRIX_ACCESS is access BOOLEAN_MATRIX;

------------------------------
-- LONG_DURATION
------------------------------
subtype LONG_DURATION is SMT_CALENDAR.LONG_DURATION;

    function "+"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."+";
    function "-"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."-";
    function "abs"(RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."abs";
    function "+"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."+";
    function "-"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return LONG_DURATION
               renames SMT_CALENDAR."-";
    function "*"(LEFT: LONG_DURATION; RIGHT: INTEGER) return LONG_DURATION
               renames SMT_CALENDAR."*";
--    function "*"(LEFT: INTEGER; RIGHT: LONG_DURATION) return LONG_DURATION
--               renames SMT_CALENDAR.MUL;
    function "/"(LEFT: LONG_DURATION; RIGHT: INTEGER) return LONG_DURATION
               renames SMT_CALENDAR."/";

    function "="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."=";
    function "<"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."<";
    function "<="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR."<=";
    function ">"(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR.">";
    function ">="(LEFT: LONG_DURATION; RIGHT: LONG_DURATION) return BOOLEAN
               renames SMT_CALENDAR.">=";

LONG_DURATION_INITIAL_VALUE : constant LONG_DURATION := 0.0;

------------------------------
-- TIME
------------------------------
subtype TIME is SMT_CALENDAR.TIME;

    subtype YEAR_NUMBER    is INTEGER       range 1901 .. 2099;
    subtype MONTH_NUMBER   is INTEGER       range 1 .. 12;
    subtype DAY_NUMBER     is INTEGER       range 1 .. 31;
    subtype DAY_DURATION   is LONG_DURATION range 0.0 .. 86_400.0;

    function CLOCK return TIME renames SMT_CALENDAR.CLOCK;
```

Copyright per DIN 34

```
    function YEAR   (DATE: TIME) return YEAR_NUMBER renames SMT_CALENDAR.YEAR;
    function MONTH  (DATE: TIME) return MONTH_NUMBER renames SMT_CALENDAR.MONTH;
    function DAY    (DATE: TIME) return DAY_NUMBER renames SMT_CALENDAR.DAY;
    function SECONDS(DATE: TIME) return DAY_DURATION renames SMT_CALENDAR.SECONDS;
    procedure SPLIT (DATE   : in     TIME;
                     YEAR   : out    YEAR_NUMBER;
                     MONTH  : out    MONTH_NUMBER;
                     DAY    : out    DAY_NUMBER;
                     SECONDS: out    DAY_DURATION) renames SMT_CALENDAR.SPLIT;

    function TIME_OF(YEAR   : YEAR_NUMBER;
                     MONTH  : MONTH_NUMBER;
                     DAY    : DAY_NUMBER;
                     SECONDS: DAY_DURATION := 0.0) return TIME
                                    renames SMT_CALENDAR.TIME_OF;

    function "+"    (LEFT: TIME;        RIGHT: LONG_DURATION)    return TIME
                                   renames SMT_CALENDAR."+";
    function "+"    (LEFT: LONG_DURATION;    RIGHT: TIME)    return TIME
                                   renames SMT_CALENDAR."+";
    function "-"    (LEFT: TIME;        RIGHT: LONG_DURATION)    return TIME
                                   renames SMT_CALENDAR."-";
    function "-"    (LEFT: TIME;        RIGHT: TIME)    return LONG_DURATION
                                   renames SMT_CALENDAR."-";

    function "="    (LEFT, RIGHT: TIME) return BOOLEAN
                                   renames SMT_CALENDAR."=";
    function "<"    (LEFT, RIGHT: TIME) return BOOLEAN
                                   renames SMT_CALENDAR."<";
    function "<="   (LEFT, RIGHT: TIME) return BOOLEAN
                                   renames SMT_CALENDAR."<=";
    function ">"    (LEFT, RIGHT: TIME) return BOOLEAN
                                   renames SMT_CALENDAR.">";
    function ">="   (LEFT, RIGHT: TIME) return BOOLEAN
                                   renames SMT_CALENDAR.">=";

    function GET_MINFRAME_INTERVAL return LONG_DURATION
                                  renames SMT_CALENDAR.GET_MINFRAME_INTERVAL;

TIME_INITIAL_VALUE : constant TIME := TIME_OF(1994, 1, 1, 0.0); --1.1.1994 why not

------------------------------
-- STATE_CODE
------------------------------
type STATE_CODE is new DB_DEFINITIONS.STATE_CODE;
STATE_CODE_INITIAL_VALUE : constant STATE_CODE :=  (others => ' ');

------------------------------
-- PULSE
------------------------------
type PULSE is new BOOLEAN;
PULSE_INITIAL_VALUE : constant PULSE := TRUE;
PULSE_NOT_TRIGGERED : constant PULSE := FALSE;
PULSE_TRIGGERED     : constant PULSE := TRUE;
```

Copyright per DIN 34

```
------------------------------
-- BURST_PULSE
------------------------------
type BURST_PULSE is new NUMERIC_TYPES.UNSIGNED_INTEGER32;
BURST_PULSE_INITIAL_VALUE : constant BURST_PULSE := 0;
BURST_PULSE_NOT_TRIGGERED : constant BURST_PULSE := 0;

------------------------------
-- BYTE_STREAM
------------------------------
type BYTE_STREAM (MAX_LENGTH : UNSIGNED_INTEGER) is
record
   ACTUAL  : UNSIGNED_INTEGER;
   ELEMENTS: UNSIGNED_BYTE_VECTOR (1 .. MAX_LENGTH);
end record;

type BYTE_STREAM_ACCESS is access BYTE_STREAM;
-- several functions concerning BYTE_STREAM

BYTE_STREAM_INITIAL_VALUE : constant BYTE_STREAM_ACCESS := null;

  type IO_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
  record
     case SW_TYPE is
       when NONE => null;
       when UNSIGNED_BYTE_TYPE =>
              UNSIGNED_BYTE_VALUE: UNSIGNED_BYTE := UNSIGNED_BYTE_INITIAL_VALUE;
       when SIGNED_BYTE_TYPE =>
              SIGNED_BYTE_VALUE:  SIGNED_BYTE := SIGNED_BYTE_INITIAL_VALUE;
       when UNSIGNED_SHORT_WORD_TYPE => UNSIGNED_SHORT_WORD_VALUE:
              UNSIGNED_SHORT_WORD := UNSIGNED_SHORT_WORD_INITIAL_VALUE;
       when SIGNED_SHORT_WORD_TYPE => SIGNED_SHORT_WORD_VALUE:
              SIGNED_SHORT_WORD := SIGNED_SHORT_WORD_INITIAL_VALUE;
       when UNSIGNED_INTEGER_TYPE => UNSIGNED_INTEGER_VALUE:
              UNSIGNED_INTEGER := UNSIGNED_INTEGER_INITIAL_VALUE;
       when SIGNED_INTEGER_TYPE =>
              SIGNED_INTEGER_VALUE: SIGNED_INTEGER := SIGNED_INTEGER_INITIAL_VALUE;
       when LONG_REAL_TYPE =>
              LONG_REAL_VALUE: LONG_REAL := LONG_REAL_INITIAL_VALUE;
       when COMPLEX_TYPE =>
              COMPLEX_VALUE: COMPLEX := COMPLEX_INITIAL_VALUE;
       when BOOLEAN_TYPE =>
              BOOLEAN_VALUE: BOOLEAN := BOOLEAN_INITIAL_VALUE;
       when TIME_TYPE =>
              TIME_VALUE: TIME := TIME_INITIAL_VALUE;
       when LONG_DURATION_TYPE =>
              LONG_DURATION_VALUE: LONG_DURATION := LONG_DURATION_INITIAL_VALUE;
       when UNSIGNED_BYTE_VECTOR_TYPE =>
              UNSIGNED_BYTE_VECTOR_VALUE: UNSIGNED_BYTE_VECTOR_ACCESS := null;
       when SIGNED_BYTE_VECTOR_TYPE => SIGNED_BYTE_VECTOR_VALUE:
              SIGNED_BYTE_VECTOR_ACCESS := null;
       when UNSIGNED_SHORT_WORD_VECTOR_TYPE => UNSIGNED_SHORT_WORD_VECTOR_VALUE:
              UNSIGNED_SHORT_WORD_VECTOR_ACCESS := null;
```

Copyright per DIN 34

```
      when SIGNED_SHORT_WORD_VECTOR_TYPE => SIGNED_SHORT_WORD_VECTOR_VALUE:
             SIGNED_SHORT_WORD_VECTOR_ACCESS := null;
      when UNSIGNED_INTEGER_VECTOR_TYPE => UNSIGNED_INTEGER_VECTOR_VALUE:
             UNSIGNED_INTEGER_VECTOR_ACCESS := null;
      when SIGNED_INTEGER_VECTOR_TYPE => SIGNED_INTEGER_VECTOR_VALUE:
             SIGNED_INTEGER_VECTOR_ACCESS := null;
      when REAL_VECTOR_TYPE => REAL_VECTOR_VALUE : REAL_VECTOR_ACCESS := null;
      when LONG_REAL_VECTOR_TYPE => LONG_REAL_VECTOR_VALUE:
             LONG_REAL_VECTOR_ACCESS := null;
      when COMPLEX_VECTOR_TYPE => COMPLEX_VECTOR_VALUE:
             COMPLEX_VECTOR_ACCESS := null;
      when BOOLEAN_VECTOR_TYPE => BOOLEAN_VECTOR_VALUE:
             BOOLEAN_VECTOR_ACCESS := null;
      when UNSIGNED_BYTE_MATRIX_TYPE =>
             UNSIGNED_BYTE_MATRIX_VALUE: UNSIGNED_BYTE_MATRIX_ACCESS := null;
      when SIGNED_BYTE_MATRIX_TYPE => SIGNED_BYTE_MATRIX_VALUE:
             SIGNED_BYTE_MATRIX_ACCESS := null;
      when UNSIGNED_SHORT_WORD_MATRIX_TYPE => UNSIGNED_SHORT_WORD_MATRIX_VALUE:
             UNSIGNED_SHORT_WORD_MATRIX_ACCESS := null;
      when SIGNED_SHORT_WORD_MATRIX_TYPE => SIGNED_SHORT_WORD_MATRIX_VALUE:
             SIGNED_SHORT_WORD_MATRIX_ACCESS := null;
      when UNSIGNED_INTEGER_MATRIX_TYPE => UNSIGNED_INTEGER_MATRIX_VALUE:
             UNSIGNED_INTEGER_MATRIX_ACCESS := null;
      when SIGNED_INTEGER_MATRIX_TYPE => SIGNED_INTEGER_MATRIX_VALUE:
             SIGNED_INTEGER_MATRIX_ACCESS := null;
      when REAL_MATRIX_TYPE =>
             REAL_MATRIX_VALUE : REAL_MATRIX_ACCESS := null;
      when LONG_REAL_MATRIX_TYPE => LONG_REAL_MATRIX_VALUE:
             LONG_REAL_MATRIX_ACCESS := null;
      when COMPLEX_MATRIX_TYPE => COMPLEX_MATRIX_VALUE:
             COMPLEX_MATRIX_ACCESS := null;
      when BOOLEAN_MATRIX_TYPE => BOOLEAN_MATRIX_VALUE:
             BOOLEAN_MATRIX_ACCESS := null;
      when REAL_TYPE =>
             REAL_VALUE : REAL := REAL_INITIAL_VALUE;
      when STATE_CODE_TYPE =>
             STATE_CODE_VALUE: STATE_CODE := STATE_CODE_INITIAL_VALUE;
      when PULSE_TYPE =>
             PULSE_VALUE: PULSE := PULSE_INITIAL_VALUE;
      when BURST_PULSE_TYPE =>
             BURST_PULSE_VALUE : BURST_PULSE := BURST_PULSE_INITIAL_VALUE;
      when BYTE_STREAM_TYPE =>
             BYTE_STREAM_VALUE : BYTE_STREAM_ACCESS := null;
      when LAST_TYPE => null;
    end case;
  end record;
```

Copyright per DIN 34

```
type IO_SCALAR_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
record
   case SW_TYPE is
     when NONE => null;
     when UNSIGNED_BYTE_TYPE =>
          UNSIGNED_BYTE_VALUE: UNSIGNED_BYTE := UNSIGNED_BYTE_INITIAL_VALUE;
     when SIGNED_BYTE_TYPE =>
          SIGNED_BYTE_VALUE:  SIGNED_BYTE := SIGNED_BYTE_INITIAL_VALUE;
     when UNSIGNED_SHORT_WORD_TYPE => UNSIGNED_SHORT_WORD_VALUE:
          UNSIGNED_SHORT_WORD := UNSIGNED_SHORT_WORD_INITIAL_VALUE;
     when SIGNED_SHORT_WORD_TYPE => SIGNED_SHORT_WORD_VALUE:
          SIGNED_SHORT_WORD := SIGNED_SHORT_WORD_INITIAL_VALUE;
     when UNSIGNED_INTEGER_TYPE => UNSIGNED_INTEGER_VALUE:
         UNSIGNED_INTEGER := UNSIGNED_INTEGER_INITIAL_VALUE;
     when SIGNED_INTEGER_TYPE =>
          SIGNED_INTEGER_VALUE: SIGNED_INTEGER := SIGNED_INTEGER_INITIAL_VALUE;
     when LONG_REAL_TYPE =>
          LONG_REAL_VALUE: LONG_REAL := LONG_REAL_INITIAL_VALUE;
     when COMPLEX_TYPE =>
          COMPLEX_VALUE: COMPLEX := COMPLEX_INITIAL_VALUE;
     when BOOLEAN_TYPE =>
          BOOLEAN_VALUE: BOOLEAN := BOOLEAN_INITIAL_VALUE;
     when TIME_TYPE =>
          TIME_VALUE: TIME := TIME_INITIAL_VALUE;
     when LONG_DURATION_TYPE =>
          LONG_DURATION_VALUE: LONG_DURATION := LONG_DURATION_INITIAL_VALUE;
     when REAL_TYPE =>
          REAL_VALUE : REAL := REAL_INITIAL_VALUE;
     when STATE_CODE_TYPE =>
          STATE_CODE_VALUE: STATE_CODE := STATE_CODE_INITIAL_VALUE;
     when PULSE_TYPE =>
          PULSE_VALUE: PULSE := PULSE_INITIAL_VALUE;
     when BURST_PULSE_TYPE =>
          BURST_PULSE_VALUE : BURST_PULSE := BURST_PULSE_INITIAL_VALUE;
     when others => null;
   end case;
end record;
```

Copyright per DIN 34

```
   type IO_COMPOSITE_VALUE_TYPE (SW_TYPE : CSS_SW_TYPES := NONE) is
   record
      case SW_TYPE is
        when UNSIGNED_BYTE_VECTOR_TYPE =>
              UNSIGNED_BYTE_VECTOR_VALUE: UNSIGNED_BYTE_VECTOR_ACCESS := null;
        when SIGNED_BYTE_VECTOR_TYPE => SIGNED_BYTE_VECTOR_VALUE:
                    SIGNED_BYTE_VECTOR_ACCESS := null;
        when UNSIGNED_SHORT_WORD_VECTOR_TYPE => UNSIGNED_SHORT_WORD_VECTOR_VALUE:
                    UNSIGNED_SHORT_WORD_VECTOR_ACCESS := null;
        when SIGNED_SHORT_WORD_VECTOR_TYPE => SIGNED_SHORT_WORD_VECTOR_VALUE:
                   SIGNED_SHORT_WORD_VECTOR_ACCESS := null;
        when UNSIGNED_INTEGER_VECTOR_TYPE => UNSIGNED_INTEGER_VECTOR_VALUE:
                   UNSIGNED_INTEGER_VECTOR_ACCESS := null;
        when SIGNED_INTEGER_VECTOR_TYPE => SIGNED_INTEGER_VECTOR_VALUE:
                    SIGNED_INTEGER_VECTOR_ACCESS := null;
        when REAL_VECTOR_TYPE => REAL_VECTOR_VALUE : REAL_VECTOR_ACCESS := null;
        when LONG_REAL_VECTOR_TYPE => LONG_REAL_VECTOR_VALUE:
                    LONG_REAL_VECTOR_ACCESS := null;
        when COMPLEX_VECTOR_TYPE => COMPLEX_VECTOR_VALUE:
                  COMPLEX_VECTOR_ACCESS := null;
        when BOOLEAN_VECTOR_TYPE => BOOLEAN_VECTOR_VALUE:
                  BOOLEAN_VECTOR_ACCESS := null;
        when UNSIGNED_BYTE_MATRIX_TYPE =>
              UNSIGNED_BYTE_MATRIX_VALUE: UNSIGNED_BYTE_MATRIX_ACCESS := null;
        when SIGNED_BYTE_MATRIX_TYPE => SIGNED_BYTE_MATRIX_VALUE:
                    SIGNED_BYTE_MATRIX_ACCESS := null;
        when UNSIGNED_SHORT_WORD_MATRIX_TYPE => UNSIGNED_SHORT_WORD_MATRIX_VALUE:
                    UNSIGNED_SHORT_WORD_MATRIX_ACCESS := null;
        when SIGNED_SHORT_WORD_MATRIX_TYPE => SIGNED_SHORT_WORD_MATRIX_VALUE:
                  SIGNED_SHORT_WORD_MATRIX_ACCESS := null;
        when UNSIGNED_INTEGER_MATRIX_TYPE => UNSIGNED_INTEGER_MATRIX_VALUE:
                   UNSIGNED_INTEGER_MATRIX_ACCESS := null;
        when SIGNED_INTEGER_MATRIX_TYPE => SIGNED_INTEGER_MATRIX_VALUE:
                    SIGNED_INTEGER_MATRIX_ACCESS := null;
        when REAL_MATRIX_TYPE => REAL_MATRIX_VALUE : REAL_MATRIX_ACCESS := null;
        when LONG_REAL_MATRIX_TYPE => LONG_REAL_MATRIX_VALUE:
                    LONG_REAL_MATRIX_ACCESS := null;
        when COMPLEX_MATRIX_TYPE => COMPLEX_MATRIX_VALUE:
                  COMPLEX_MATRIX_ACCESS := null;
        when BOOLEAN_MATRIX_TYPE => BOOLEAN_MATRIX_VALUE:
                  BOOLEAN_MATRIX_ACCESS := null;
        when BYTE_STREAM_TYPE =>
                BYTE_STREAM_VALUE : BYTE_STREAM_ACCESS := null;
        when others => null;
      end case;
   end record;

end CSS_TYPES;
```

Copyright per DIN 34

6.5 Model Observation & Control

6.5.1 Basics

The **M**odel **O**bservation and **C**ontrol **S**ystem (MOCS) has to accomplish two main task:

* CSS system commanding and
* CSS system state display

CSS system commanding includes the commanding of the Kernel (the simulator) and the definition of the system presentation (on–line definition for monitoring, logging and tracing).

CSS system observation functions provide the display of the states of e.g. the MOCS users, MOCS itself, the Kernel, the **I**mmediate **C**ommand **P**rocessor (ICP) and the monitored values.

CSS distinguishes two execution phases: the development phase and the operational phase.

Mocs can be started:

* by I_MDB during the ***development phase*** (this is described later in this document)
  Commands can be entered graphically from a MOCS main window or additional monitoring windows.
* by VICOS/HCI–ICP during the ***operational phase*** (MOCS starts up in real time mode and automatically accepts connections with ICP).
  Commands can be entered as a textual command string in a HLCL command window which is controlled either locally by ICP SW or remotely by VICOS SW. (The available commands are part of the HLCL description.)

Note that this chapter concentrates on the graphical commanding of MOCS (the development phase). The commanding of CSS during the operational phase is not part of this chapter.

The following steps describe in a short form how to proceed to get a simulation running (during the development phase).
During model editing the user should perform preliminary tests to check wether the model works according to the implementation goals or not.

1. Select the desired model in the I_MDB window and choose **MOCS**.
2. **Start** or **Connect** the simulator in the Simulation Controller window (simulator subview).
3. Select the host machine for the kernel.
4. Select the simulation state and **Load** it in the Simulation Controller (simulation subview).
5. Push the **Open Observer** button in the Simulation Controller (simulator subview) and select a simulation table. Open the simulation table in the Session Observer window.
6. Push the **Set** simulation mode button in the Simulation Controller (simulation subview) and set the simulation parameters.
7. **Start** the simulation in the Simulation Controller (simulation subview).
8. Observe the simulation results and messages in the Session Observer window and the MOCS Console window.

Copyright per DIN 34

9. Manipulate input variables according to the testing objective.
10. **Stop** the simulation in the Simulation Controller (simulation subview).
11. **Stop** or **Disconnect** the simulator in the Simulation Controller (simulator subview).
12. Quit the CSS model evaluation session.
13. Now the logging files are available for inspection.

Detailed informations about the individual steps are given in the next sections.

6.5.2 Starting MOCS from I_MDB

The CSS invocation function provided in I_MDB allows to start the CSS tool. This functions can ***only*** be called up when the user is within a ***CCU scope***.

After a model has been successfully configured you can start MOCS to execute the model.

A description how to navigate in the data base is given in the procedures in section 6.4.4.1. They describe in detail how to start I_MDB, how to select CCU and CDU and how to start CSS from I_MDB.

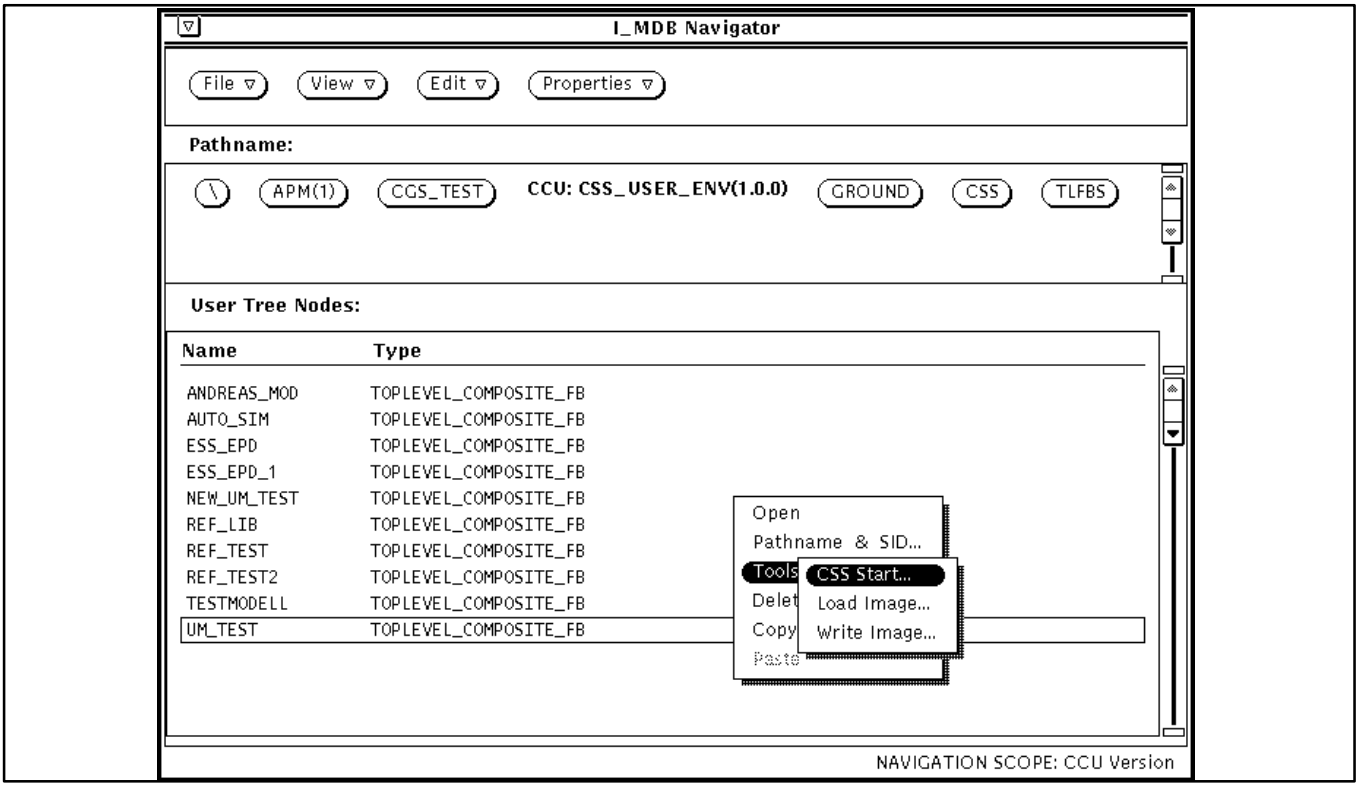


Figure 6–87 : *The CDU contains several models*

The preconditions for the following procedure are:

* a CCU is selected which contains all available references
* the CDU which contains the desired model is open

Copyright per DIN 34

Start the CSS User Interface for Model execution

Start the CSS User Interface for Model execution

- **Select** the model in the I_MDB window. (see Figure 6–87)
- Press the right mouse button and select **Tools –> CSS Start...** from the pop–up menu.
- A confirmation window pops up. Press the **Ok** button.
- The CSS scope check window appears. MDE is pre–selected. Press the **MOCS** button. (see Figure 6–88)
- Press the **Start CSS...** button.
- The Database Browser window and the Simulation Controller window will be opened automatically.

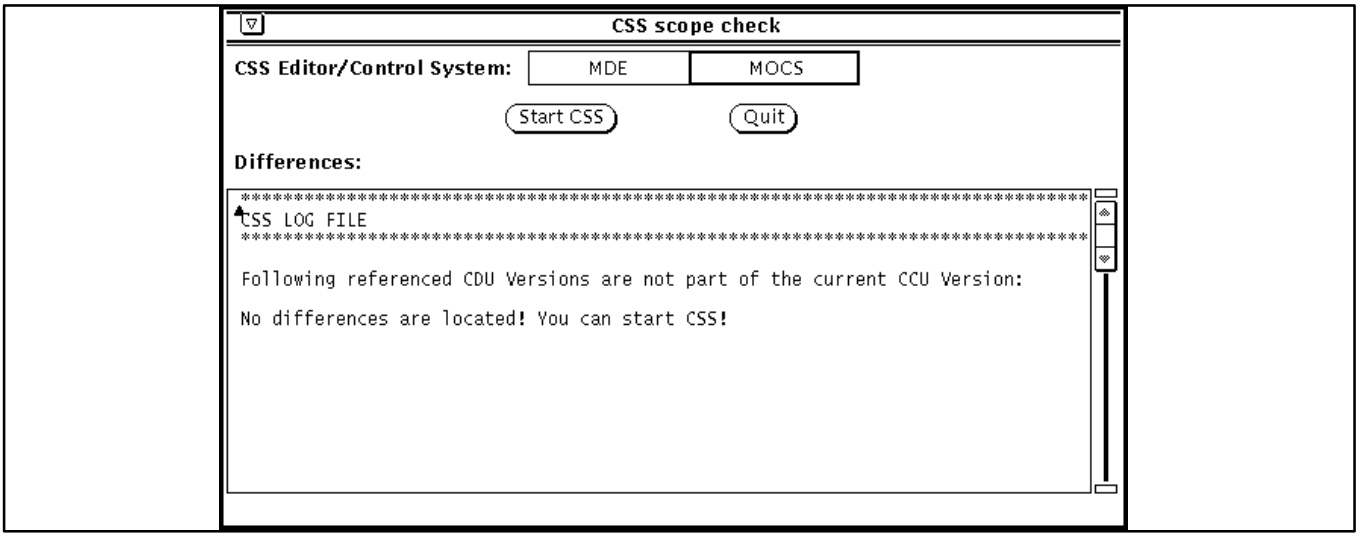


Figure 6–88 : *Select MOCS in the CSS scope window to start the model execution*

The CSS Simulation Controller window appears (additionally the DBB window pops up).

Copyright per DIN 34

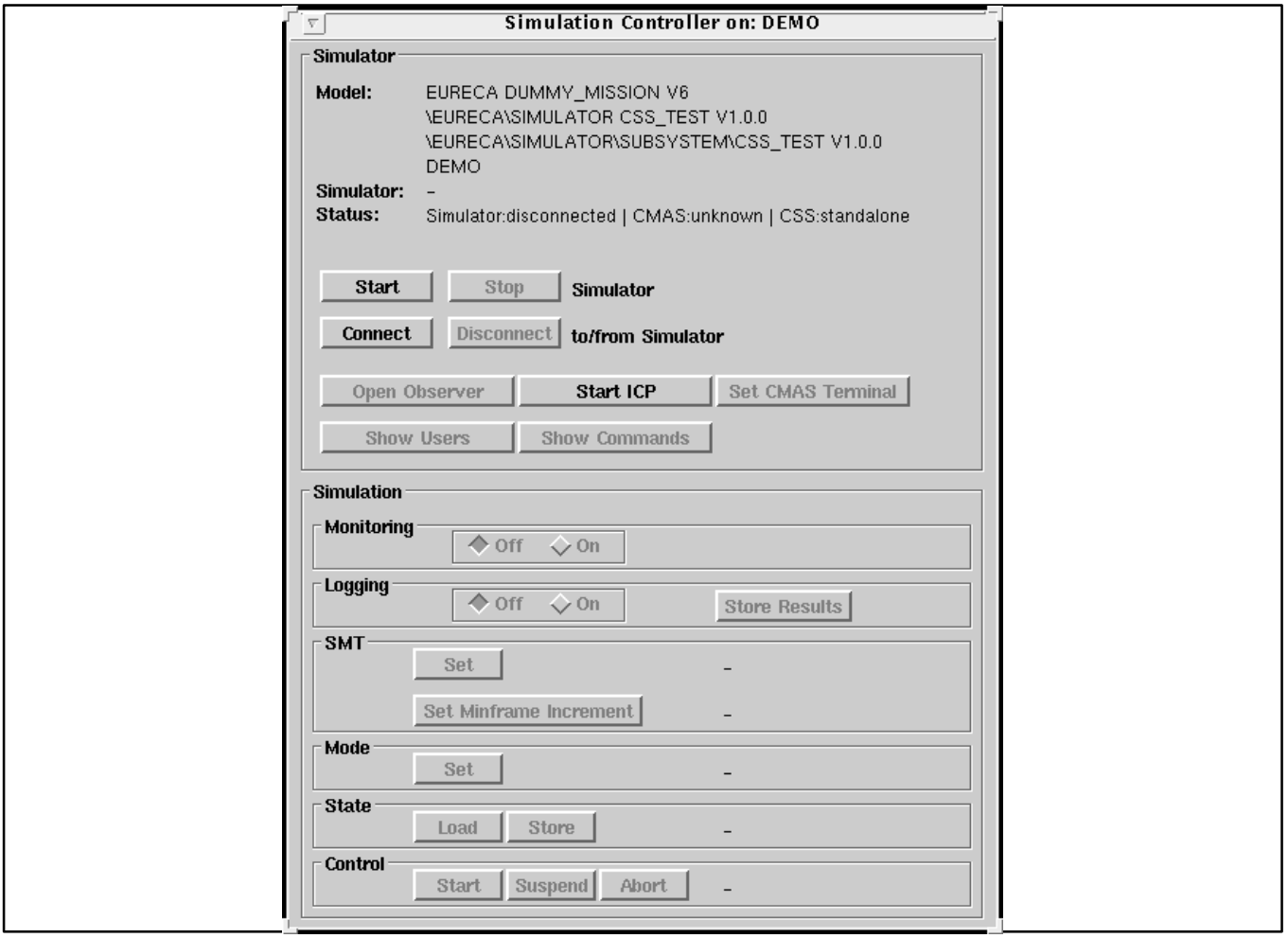


Figure 6–89 : *The initial CSS Simulation Controller window*

The window shows the following informations:

* the **model identification**
  the element, mission identification, system_tree_version,
  the CCU with version, issue and revision,
  the CDU with version, issue and revision
  and the model name
* the **simulation session status**,
  defaults are: **disconnected** for the simulator, **unknown** for CMAS and **standalone** for the CSS configuration (if CSS was called from I_MDB)

The CSS Simulation Controller window consists of two parts. The upper part (simulator subview) commands the start and stop of the kernel (i.e. the simulator), the connection/suspension to/from an already running simulator. Buttons give access to additional functions, e.g. the predefined simulation tables, a list of already running simulators. With the help of these function the user prepares his/her simulation environment.

The lower window part (simulation subview) contains the buttons to control the currently running simulation session. The user selects the simulation mode, i.e. stepwise or continuous model execution and uses buttons to control the simulation execution (start, suspend and abort).

Copyright per DIN 34

6.5.3 Starting the simulator

The buttons in the upper part (simulator subview, see Figure Figure 6–89) have the following functions:

* the **Start** *Simulator* button
  starts the Kernel (simulator) on a machine of your choice.
* the **Stop** *Simulator* button
  stops the Kernel.
* the **Connect** *to/from Simulator* button
  connects the user to a pre–selected and already running Kernel (simulator).
* the **Disconnect** *to/from Simulator* button
  disconnects the user from the simulator. The Kernel stays active.
* the **Open Observer** button
  opens a separate window where you can load and activate a simulation table and make on–line changes
* the **Show Users** button
  displays a list of all users connected to a simulator and is the place, where user privileges are requested and withdrawn
* the **Show Commands** button
  shows all pending commands in a separate window (see Figure 6–91)

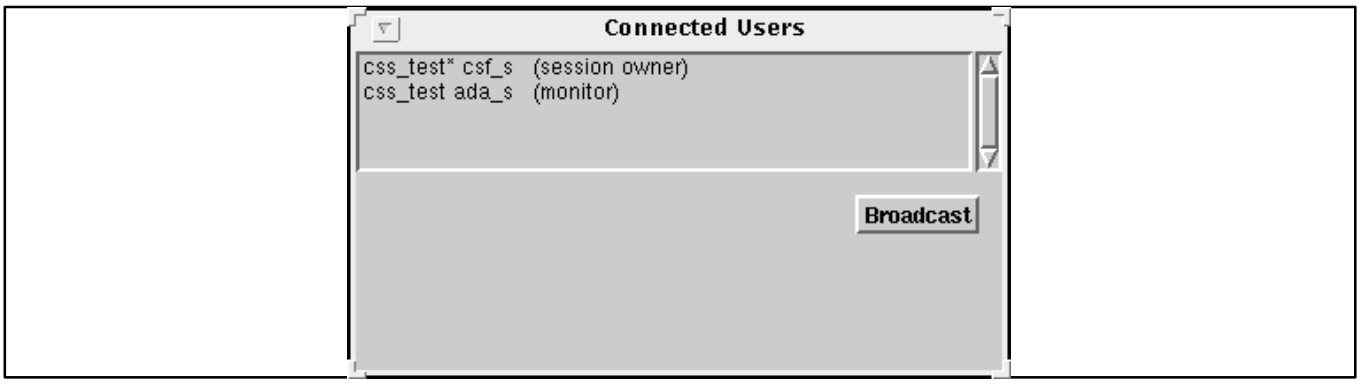


Figure 6–90 : *The Connected Users window shows a list of connected users and the status*

* the **Set CMAS Terminal** button
  is normally disabled. It will be activate if CMAS is running. The button allows to switch of remote terminals.
* the **Start ICP** button
  opens the ICP input window and starts the ICP. The ICP (Immediate Command Processor) window serves as an interactive gateway to the CSS commanding via HLCL.

Copyright per DIN 34

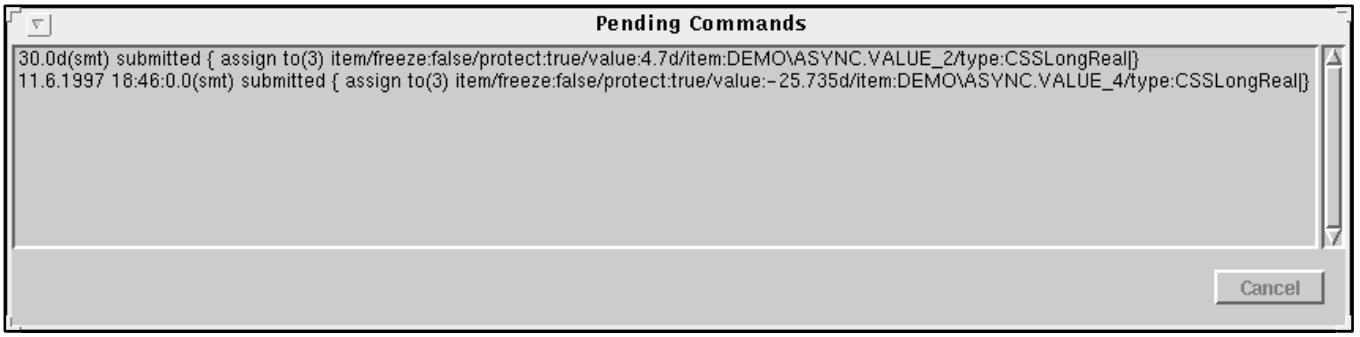

Pending Commands
30.0d(smt) submitted { assign to(3) item/freeze:false/protect:true/value:4.7d/item:DEMO\ASYNC.VALUE_2/type:CSSLongReal|}
11.6.1997 18:46:0.0(smt) submitted { assign to(3) item/freeze:false/protect:true/value:-25.735d/item:DEMO\ASYNC.VALUE_4/type:CSSLongReal|}
Cancel

Figure 6–91 : *The pending commands window shows the commands and the affected model items*

Starting the Simulator

- Press the **Start** *Simulator* button using the left mouse button. The **Start Simulator** window pops up. (see Figure 6–92)
- Type the machine name in the **Kernel Host** field (see Figure 6–92). The actual name of the host machine depends on the HW environment where you are working.
- Press the **Apply** button. The status in the status line changes accordingly.

As long as the model itself is the item under test it's not necessary to start CMAS. CMAS is needed as the connecting part to HW.

Click on the *Yes* button of the **Start CMAS** box.

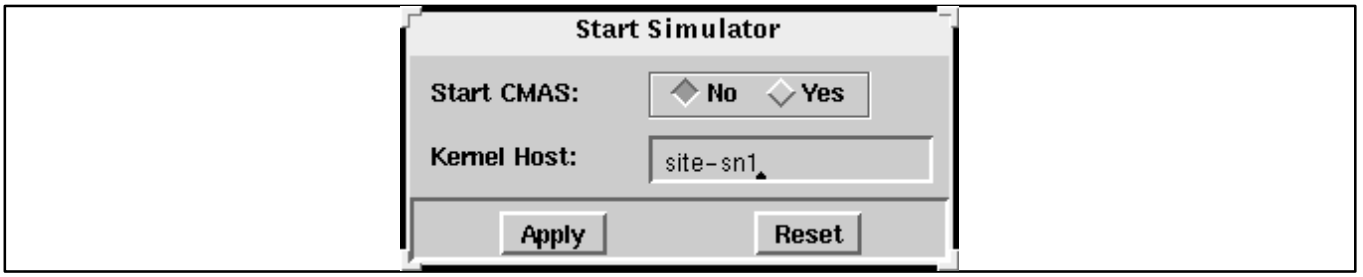


Figure 6–92 : *The Start Simulator window allows to specify the host machine*

Instead of starting the simulator every time you start a model execution it's possible to disconnect from the simulator and to re–connect at a later time. (This feature is only useful if you proceed with your work using the same model i.e. no changes were made). You leave the simulator using the **Disconnect** *to/from Simulator* button. The next time you can connect to the simulator which was still running in the meantime.

Copyright per DIN 34

Connecting to a Simulator

- *Precondition for the connection to a simulator is that a session was quit with the* **Disconnect** *to/from Simulator* button.
- Press the **Connect** *Simulator* button using the left mouse button. The **Simulators** window pops up.
- Select the simulator by clicking on the name in the list (see Figure 6–93). The name becomes highlighted.
- *Note that only those simulators are displayed which belong to the previously selected model.*
- Press the **Apply** button. The status in the status line changes accordingly.

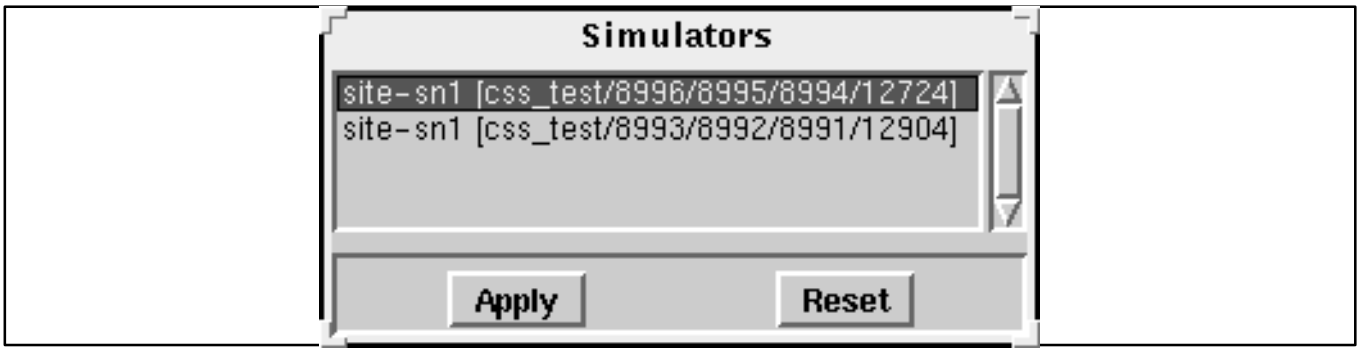


Figure 6–93 : *Select the kernel from the list of running simulators*

- *Note that the list of running simulators you see in the Simulators window belongs to just one model (the one you selected previously in I_MDB).*
  Make sure that you have finished your working task (i.e. the simulation is stopped) before working with another model.

6.5.4 User authorisation

The user who starts the simulator is called the ”session owner”. He/she is the person responsible for logging values, injecting values into the model, and finally stopping or restarting the simulator. All other users may connect to the session and may monitor values but are not allowed to control the simulation session.
The session owner can share his privileges with other users. As seen in Figure 6–94 the session owner is the user css_test working on the machine csf_s. A user working under the same user name but on a different workstation has the monitor privilege only.

Copyright per DIN 34

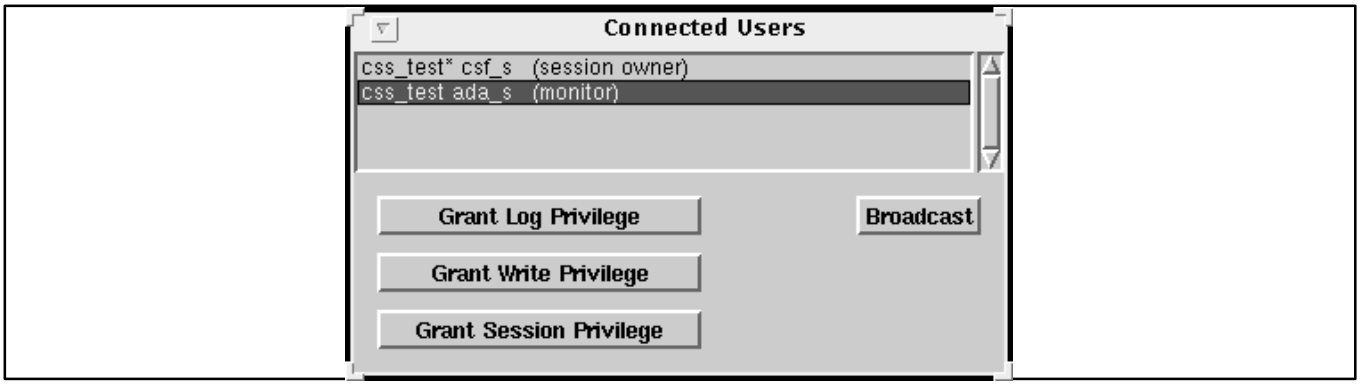


Figure 6–94 : *The selected user can receive privileges from the session owner*

To request a special privilege the user has to press the corresponding button. The message will be created automatically and send directly to the session owner (see Figure 6–95)

Figure 6–95 : *The user css_test on ada_s sends a request for the logging privilege*

If the session owner agrees to the request he/she selects the user in the Connected Users window (the user name must be highlighted) and pushes the **Grant Log Privilege** button. The entry in the Connected Users window changes accordingly (see Figure 6–96)

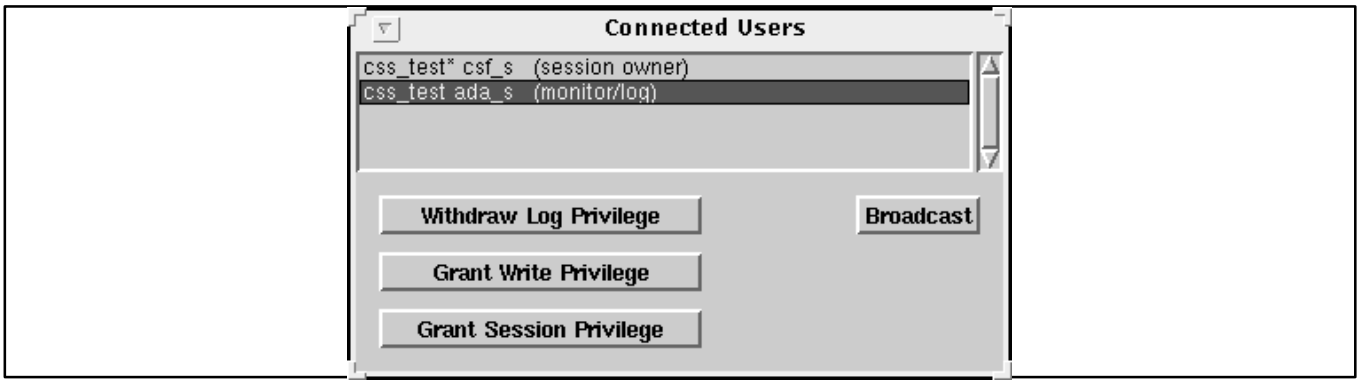


Figure 6–96 : *The user css_test on ada_s received the logging privilege.*

The session owner is allowed to withdraw given privileges.

*Important: There is always only one session owner. As soon as the session owner gives the **Session** privilege to another user he/she looses the session ownership. The user with the **Session** privilege is the session owner.*

The **broadcast** button allows to send a message to all connected users. Press the broadcast button and an input window appears.

Copyright per DIN 34

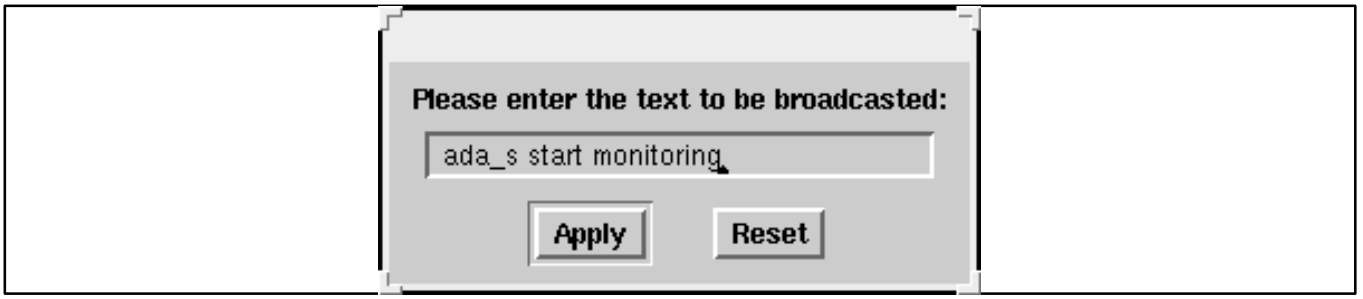


Figure 6–97 : *The broadcast message is send to **all** connected users*

Copyright per DIN 34

6.5.5 Starting a simulation session

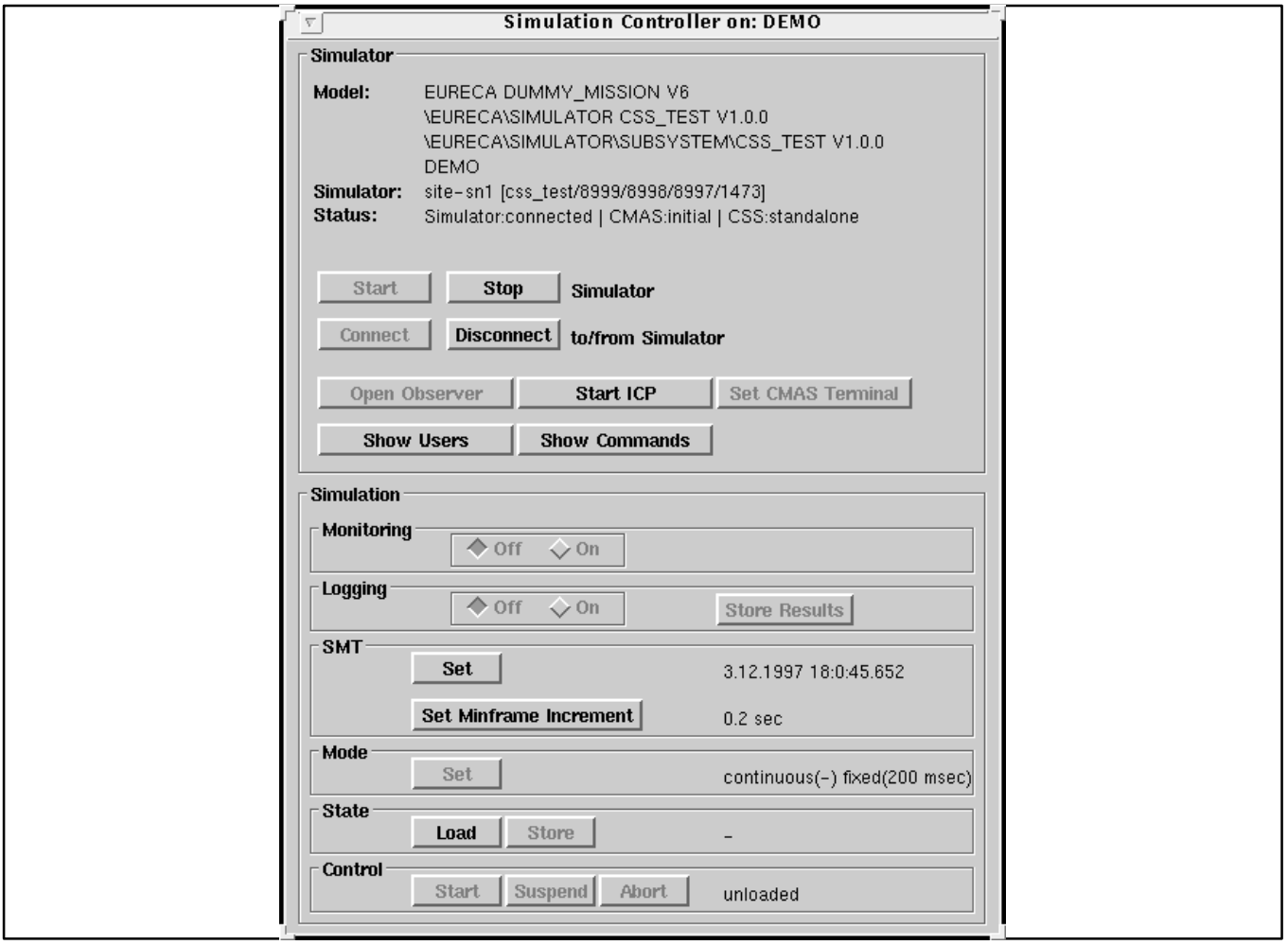


Figure 6–98 : *The CSS Simulation Controller window with running Simulator*

The buttons in the lower part (simulation subview, see Figure Figure 6–98) have the following functions:

* the **Monitoring: On** and **Monitoring: Off** buttons
  switch monitoring on or off
* the **Logging: On** and **Logging: Off** buttons
  switch logging on or off
* the **Logging: Store Results** button
  allows to store the logging results (i.e. the data accumulated since the last storing or since the start of the simulation) in the file system (i.e. create archive files and optionally data set info files and data set files)
* the **SMT: Set** button
  allows to set the Simulated Mission Time (SMT). Per default the value is set to the machine time when the kernel starts.
* the **SMT: Set Minframe Increment** button
  allows to set the duration of a minframe in SMT.

Copyright per DIN 34

* the **Mode: Set** button
  opens a separate window where you can adjust the parameters for model execution.
* the **State: Load** button
  opens a window with a list of available simulation states. The selected simulation state will be loaded.
* the **State: Store** button
  stores the current state of the simulation. In later sessions it is possible to start at that point again.
* the **Control: Start** button
  starts model execution.
* the **Control: Suspend** button
  halts model execution if you are running the model in continuous mode.
* the **Control: Abort** button
  stops model execution if the simulation is running faulty, i.e. an error condition occurred during simulation execution.

The monitoring and logging functions are performed as soon as the respective **On** button is selected and the simulation is started.

*Note that there are no **On/Off** buttons for tracing definitions. The tracing function is active as soon as the simulation is started.*

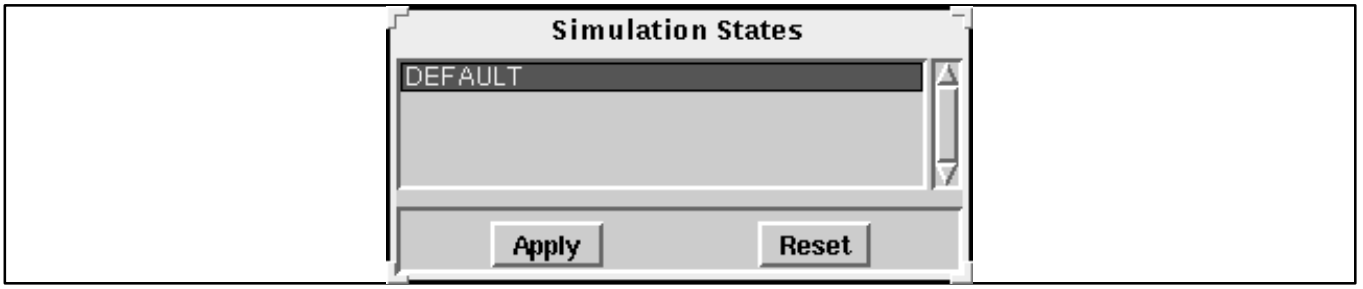


Figure 6–99 : *Select the simulation state from a list*

Copyright per DIN 34

Starting a simulation session for model testing

- Press the **Load** *State* button using the left mouse button. The **Simulation States** window pops up.
- Select a simulation state by clicking on the name in the list (see Figure 6–99). The name becomes highlighted.
- *The simulation will start execution in the state you just loaded. The DEFAULT state is the initial state for the model.*
- Press the **Apply** button. The status in the status line changes accordingly.
- Press the **Set** *Mode* button. The **Simulation Mode** window appears. (see Figure 6–100)
- *For first tests **step mode** is recommended.*
- Click in the **Stepwise** box.
- Press the **Apply** button.
- Press the **Set Minframe Increment** *SMT* button. An input window appears.
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the desired duration in the **Value** input field.
- Press the **Apply** button.
- Click on the **Monitoring On** button. The button becomes highlighted.
- Click on the **Logging On** button. The button becomes highlighted.
- Press the **Open Observer** button. The **Simulation Tables** window pops up.
- Select a simulation table from the list by clicking on its name in the list. (see Figure 6–101).
- Press the **Apply** button. *to be continued.....*

- Resize the session observer window conveniently.
- Press the **Start** *Control* button.

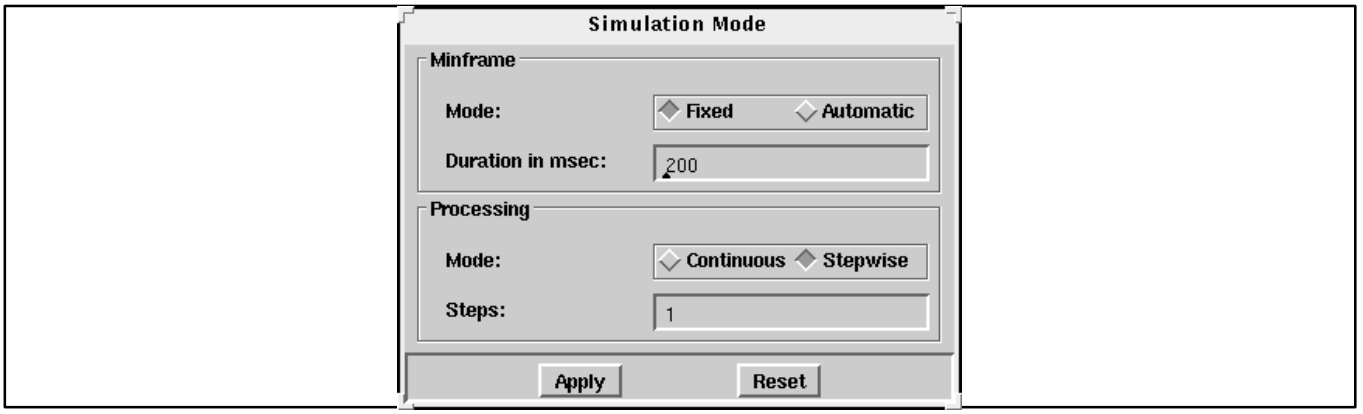


Figure 6–100 :*Setting the simulation mode parameters*

A more detailed description of the different execution modes and the timing capabilities will be given in the following sections.

Copyright per DIN 34

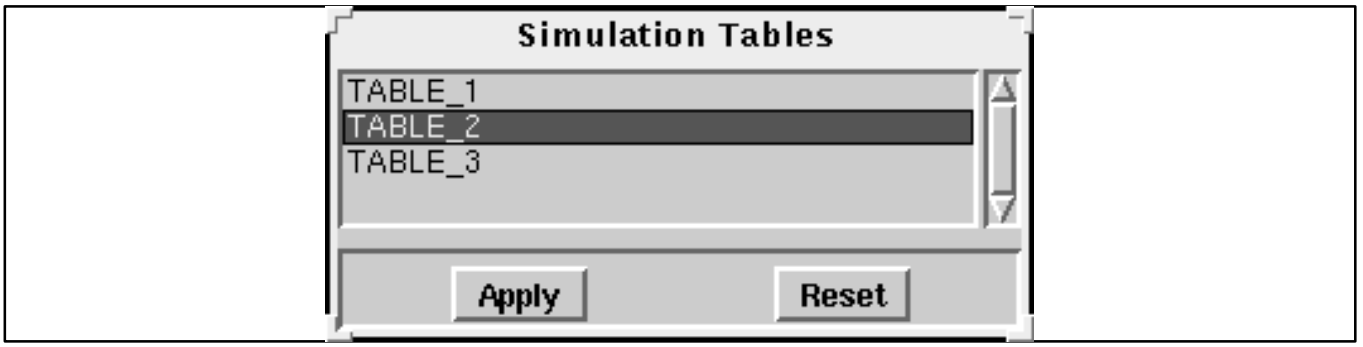


Figure 6–101 :*Select a simulation table from the list of available tables*

An overview about all testing capabilities (i.e. error injection into the model, parameter checks) will be given in the following sections.

Closing a simulation session

- If the simulation is running in continuous mode, press the **Suspend** *Control* button.
- If monitoring is enabled, click on the **Monitoring Off** button. The button is highlighted.
- If logging is enabled, click on the **Logging Off** button. The button is highlighted.
- *Perform the following steps if there are open simulation tables.*
- Press the S**tore Table** button to save changes in the table you made during the simulation session. An input window appears.
- Replace the table name by a new one if you want.
- Press the **Apply** button to save the changed table.
- Move the mouse pointer in the label of the Session Observer window and select **Quit** from the window pop–up menu. The Session Observer window disappears.
- Move the mouse pointer into the Simulation Controller window and press the **Stop** *Simulator* button.
- Move the mouse pointer in the label of the Simulation Controller window and select **Quit** from the window pop–up menu. The Simulation Controller window disappears.

Copyright per DIN 34

6.5.6 Simulation execution

6.5.6.1 The Session Observer window

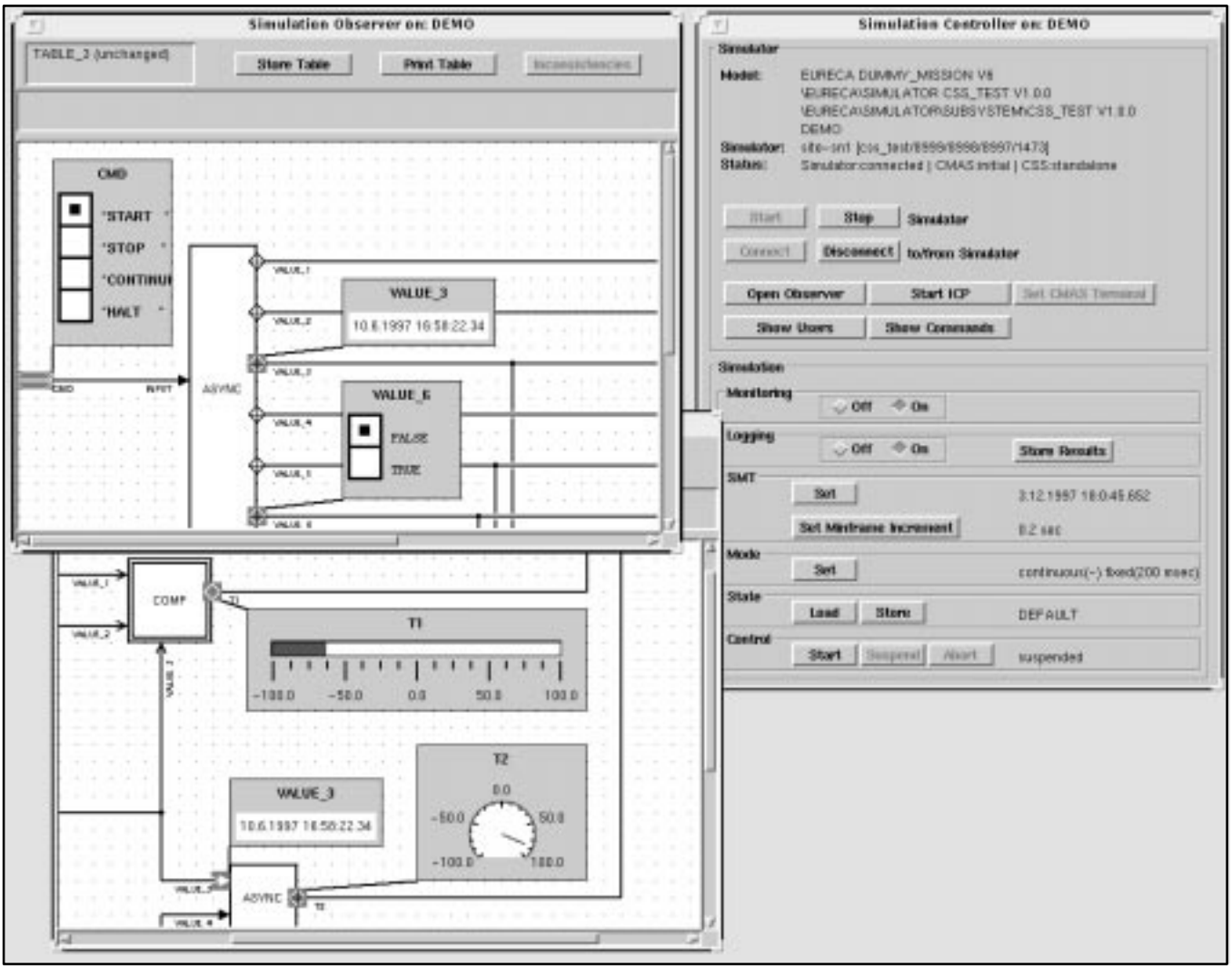


Figure 6–102 :*The Simulation Controller window with two active Session Observer windows showing different levels of the model*

6.5.6.2 Basics

The main purpose of the Model Observation and Control function of CSS is to serve as the user interface for interactive simulation control and monitoring. This is done via the **Session Observer** window. After loading a simulation state you can press the **Open Observer** button. The **Simulation Tables** window allows the selection of the desired simulation table from a list.

Note that the precondition for on–line loading and changing tables is that a simulation state is loaded. Otherwise the command button **Open Observer** *is disabled.*

It is possible to load more than one simulation table and to switch during simulation execution to another table (see Figure 6–102), tables can be changed on–line and the changes can be stored.

Copyright per DIN 34

The three buttons in the upper part (see Figure 6–103) have the following functions:

* the **Store Table** button
  stores the changes made during on–line table editing
  (either the table is updated or a new table can be created)
* the **Print Table** button
  prints the parameters and their monitoring, tracing or logging status
* the **Inconsistencies** button
  shows the inconsistent definitions with respect to the current simulation model contained in the simulation table

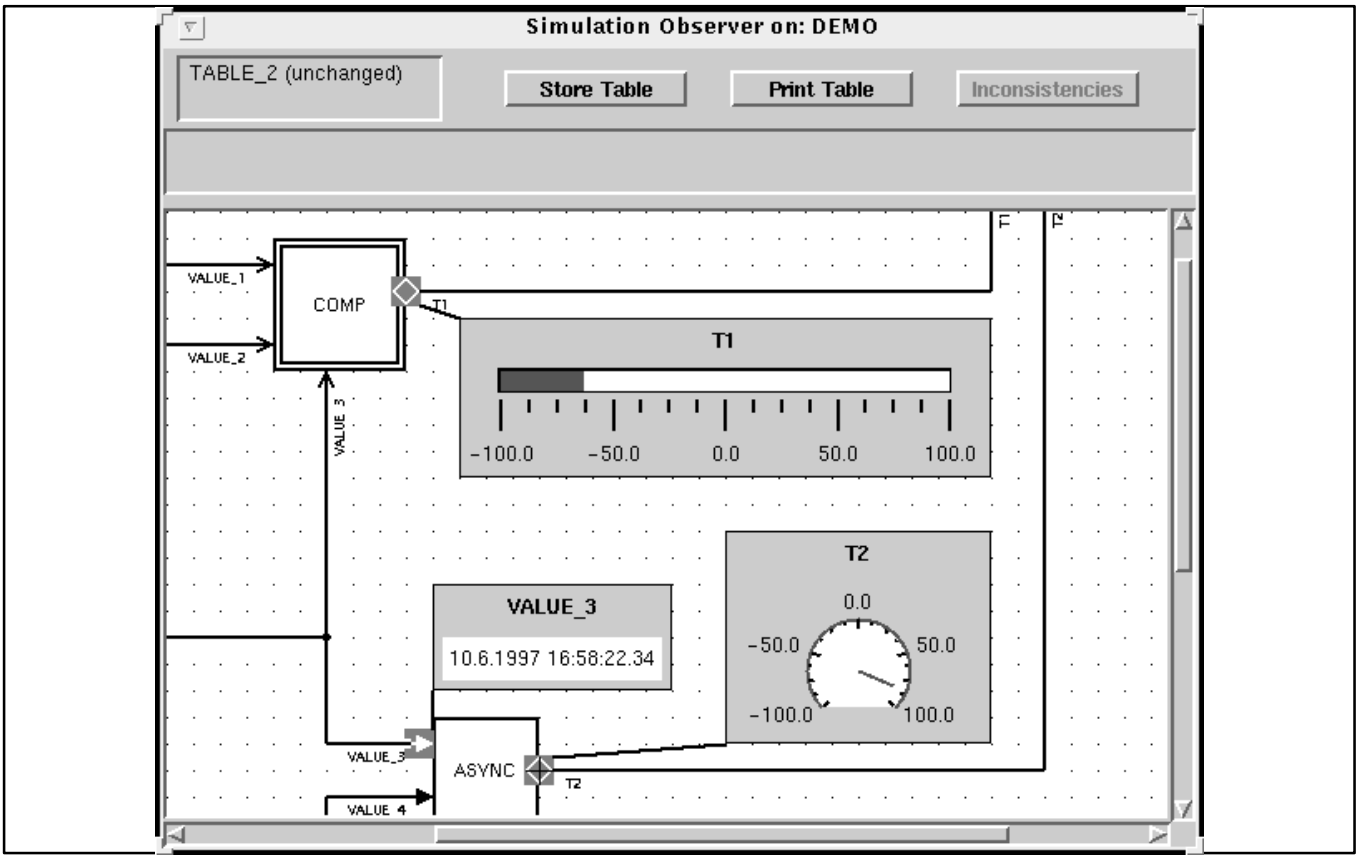


Figure 6–103 :*The Session Observer used to oberve the simulation*

6.5.6.3 On–line monitoring

All functions already described in section 6.4.7.9 are available during on–line monitoring, including the creation of new simulation tables. This can be achieved by editing an already existing simulation table and storing it with a different – currently unused – name.

A set of additional functions for on–line testing is provided

* **snapshot –> to screen**
  Display the current values of the selected atomic outputs and inputs, parameters and model inputs and outputs in the MOCS Console window.

Copyright per DIN 34

* **snapshot all –> to screen**
  Display the current values of all atomic outputs, parameters and model inputs in the MOCS Console window.

* **snapshot –> to log file**
  Write the current values of the selected atomic outputs and inputs, parameters and model inputs and outputs to an archive file (refer to section 6.4.7.9.4 for more information on logging and archive files).

* **snapshot all –> to log file**
  Write the current values of all atomic outputs, parameters and model inputs to an archive file (refer to section 6.4.7.9.4 for more information on logging and archive files).

* **snapshot –> as HLCL sequence**
  Generate a file containing HLCL statements that assign the respective current value to the selected atomic outputs and inputs, parameters and model inputs and outputs. The MOCS Console window gives the name of the file generated.

* **snapshot all –> as HLCL sequence**
  Generate a file containing HLCL statements that assign the respective current value to all atomic outputs, parameters and model inputs. The MOCS Console window gives the name of the file generated.

The HLCL sequence is stored in the Unix file system in a directory which is determined by the environment variable CSS_LOG_DIR . The file is marked with the extension .SNAPSHOT_HLCL_#, the sharp sign indicates the serial number. The part of the name preceeding the extension matches the same pattern as for log and archive files (refer to section 6.4.7.9.4 for more information on names of log and archive files).
In contrast to the simulation model's simulation states which are automatically deleted when a new simulator kernel is configured, these HLCL sequences may survive such configurations and then subsequently be used to set the values of the model variables according to a previous state (i.e. the state at snapshot time). However, the user is responsible that the HLCL sequence is still compatible to the simulation model with respect to pathnames and types of the interface items assigned. If necessary, a HLCL sequence may be edited using a text editor to adapt it to a modified simulation model.

* **freeze**
  Freeze the last value of an atomic output or model input. The value will not change during further execution.

* **reactivate**
  Reactivate a frozen atomic output or model input.

* **assign –> don‘t freeze**
  Assign a value to an atomic output (which is marked with a 'write access' marker) or model input. The value will be sent once.

* **assign –> freeze**
  Assign a value to an atomic output (which is marked with a 'write access' marker) or model input, then freeze the atomic output or model input.

Note that you are not allowed to assign values to atomic inputs or model outputs.

Copyright per DIN 34

Taking a snapshot to screen

- Select the item (input or output) in the Session Observer window by clicking on it with the left mouse button.

  It is possible to get more than one value with one snap shot command.

- Select the desired input/output by clicking on it with the middle mouse button. Repeat this until all desired items are selected.
- Press and hold the right mouse button. Select the command **snapshot–>to screen** from the pop–up menu.
  The values appear immediately in the MOCS Console window (see Figure 6–104)

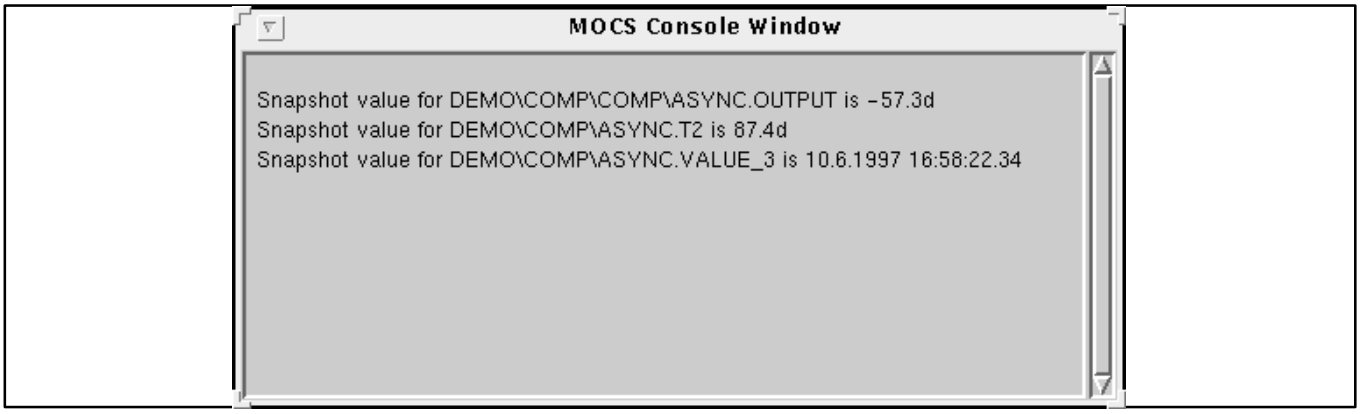

MOCS Console Window
Snapshot value for DEMO\COMP\COMP\ASYNC.OUTPUT is -57.3d
Snapshot value for DEMO\COMP\ASYNC.T2 is 87.4d
Snapshot value for DEMO\COMP\ASYNC.VALUE_3 is 10.6.1997 16:58:22.34

Figure 6–104 :*Snapshot values are displayed in the MOCS Console window*

Assign a value to an atomic output or model input

- Select the item by clicking on it with the left mouse button.
- Press and hold the right mouse button. Select **assign** from the pop–up menu. The Assign window appears (see Figure 6–105)
- Type the new value in the **Value** input field.

  Now you can freeze the item.

- Click into the **Freeze** box. The value will be valid for the item until the item is reactivated.
- Press the **Apply** button. The assign command will be performed immediately.

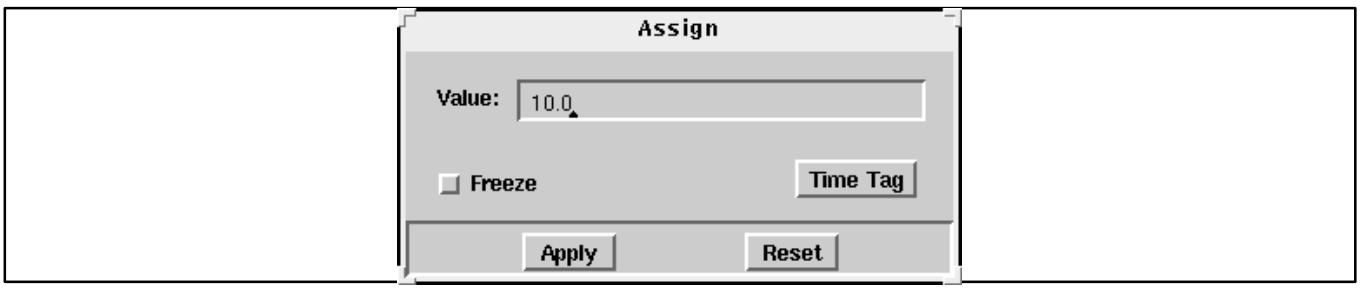


Figure 6–105 :*The Assign window*

Copyright per DIN 34

Figure 6–106 :*Specifying a time tag in SMT*

Assign a time tagged value to an atomic output or model input

- Select the item by clicking on it with the left mouse button.
- Press and hold the right mouse button. Select **assign** from the pop–up menu. The Assign window appears (see Figure 6–105)
- Type the new value in the **Value** input field.

  Now you can freeze the item.

- Click into the **Freeze** box. The value will be valid for the item until the item is reactivated.

  A time tag delays the execution of the assign command for a specified time.

- Press the **Time Tag** button. The Time Tag specification window appears (see Figure 6–106).
- Click into the **Time Specification: Time** box, then click either into the **Time Scale: SMT** or **Local Time** box.

- Type the date and time in the **Value** input field (see Figure 6–106)
  OR
- Click into the **Time Specification: Duration** box, then click either into the **Time Scale: SMT** or **Local Time** box.
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the duration in the **Value** input field (see Figure 6–107)
- Press the **Apply** button. The assign command will be performed at the specified time or after a specified duration.
- A confirmation window pops up (see Figure 6–108). Press the OK button.

Figure 6–107 :*Specifying a duration in SMT*

Copyright per DIN 34

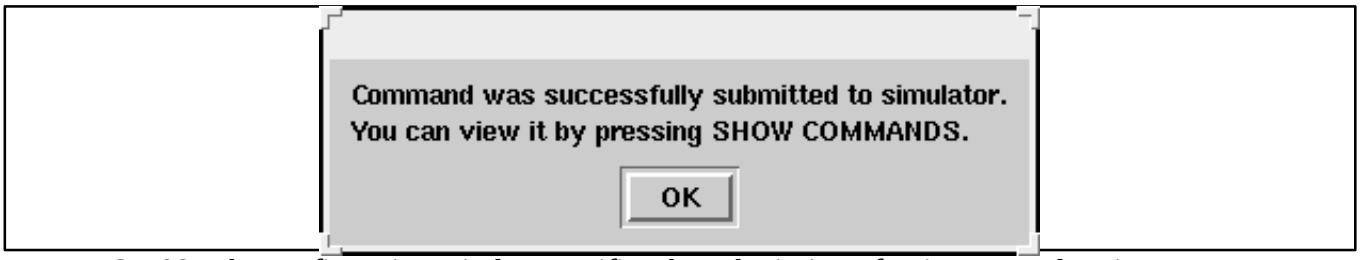


Figure 6–108 :*The confirmation window certifies the submission of a time tagged assign command*

6.5.6.4 Time scales used during simulation execution

There are two independent time scaling systems used in the simulation:

* the SMT (Simulated Mission Time) constitutes the ”official” simulation time.
  The model developer can set the SMT increment per minframe according to his needs, so that the SMT clock can show the hours passing by (and not the actual local time used for model execution); an action which takes some hours in real time can be simulated in seconds.
* the LT (Local Time) is the actual time in which simulation execution takes place.
  In fixed minframe mode, the execution of each minframe lasts exactly the specified constant duration (the minimum value is 50 msec). In automatic minframe mode, each minframe lasts exactly the time needed for performing all computations; this results in varying minframe durations.

6.5.6.4.1 Setting the Simulated Mission Time

6.5.6.4.1.1 Setting the SMT starting point

As soon as you start the simulator, CSS sets the SMT to the current date and time which is provided by the kernel host.
You can choose any time you like as starting point.

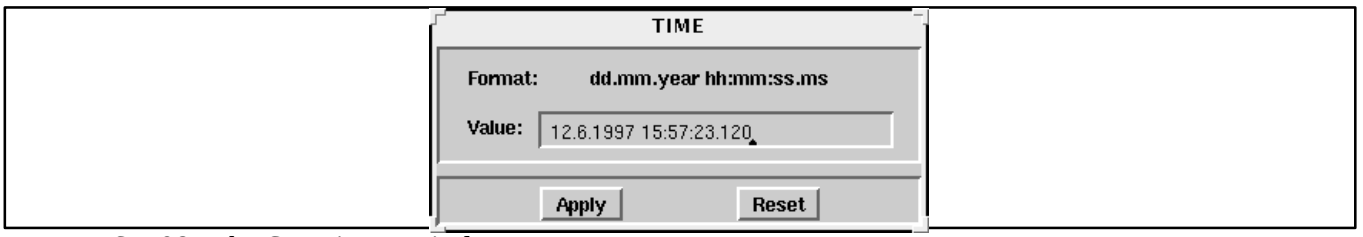


Figure 6–109 :*The SMT input window*

The following input format is expected:

dd	day – one or two digits (1 – 31)
mm	month – one or two digits (1 – 12)
year	year – must be four digits
hh	hours – one or two digits (0 – 23)
mm	minutes – one or two digits (0–59)
ss	seconds – one or two digits (0–59)

Copyright per DIN 34

ms milliseconds – up to three digits (0–999)

Day, month and year must be separated by a point; hours,minutes and seconds by a colon. Milliseconds must be separated by a point. Note that there is a blank between date and time.

Setting the SMT starting point

- If the simulation was already started, you have to load a simulation state first before you can change the time setting.
- Press the **Set** *SMT* button. The SMT input window pops up (see Figure 6–109).
- Type the desired date and time in the **Value** input field.
- Press the **Apply** button. The SMT is changed in the Simulation controller window.

6.5.6.4.1.2 Setting the SMT increment per minframe

The SMT minframe increment determines how long one minframe shall last in SMT.

Setting the SMT increment per minframe

- If the simulation was already started, you have to load a simulation state first before you can change the SMT increment per minframe.
- Press the **Set Minframe Increment** *SMT* button. The SMT increment input window pops up (see Figure 6–110).
- Click in the **Unit** box (select either **Seconds**, **Minutes** or **Hours**)
- Type the desired duration in the **Value** input field.
- Press the **Apply** button.

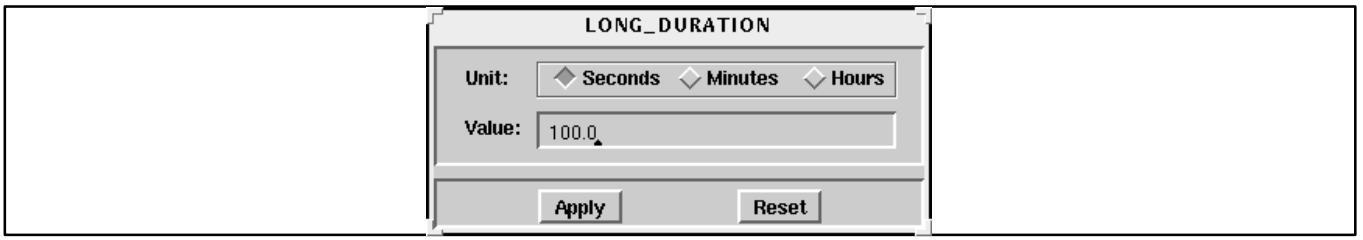


Figure 6–110 :*The SMT increment input window*

6.5.6.5 The non–Real Time Simulation Modes

Figure Figure 6–111 shows the default simulation mode setting suggesting the **Fixed** *Minframe* mode and the **Continuous** *Processing* mode.

Copyright per DIN 34

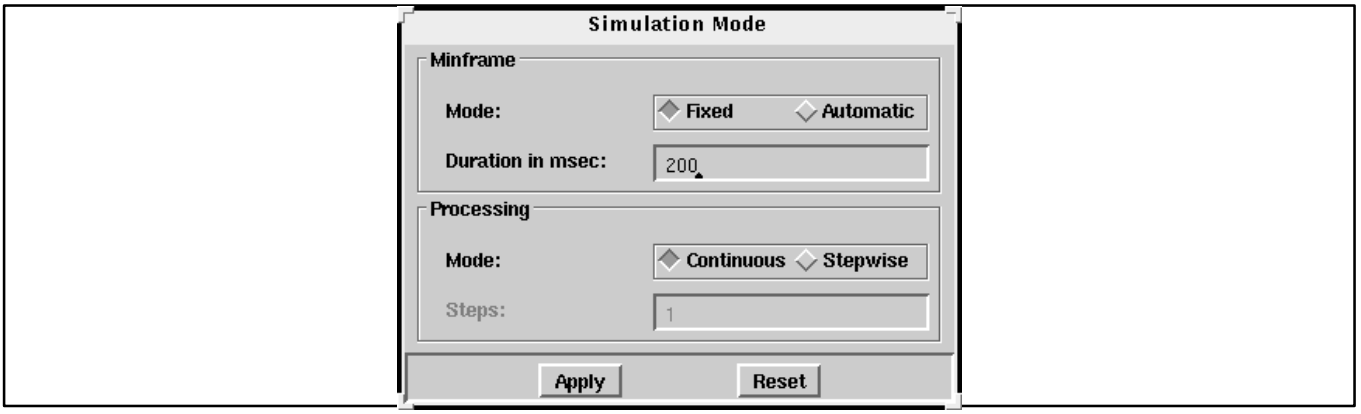


Figure 6–111 :*The Simulation Mode window in default state*

6.5.6.5.1 The Minframe Mode

In fixed minframe mode, the execution of each minframe lasts exactly a specific constant duration. You can specify this value in the **Duration in msec** input field (the minimum value is 50 msec).

The automatic minframe mode could also be called the 'as fast as possible' mode. As soon as all computations for a step (minframe) are performed, the computations for the next step (minframe) are started. The minframe duration is determined by the amount of time needed to perform all calculations for a step and varies from step to step.

6.5.6.5.2 The Processing Mode

You can choose between either the continuous or the stepwise processing mode. For first checks stepwise processing mode is recommended. You can type the number of steps, i.e. minframes to be executed when you start the simulation, in the **Steps** input field. If you start the simulation by pressing the **Start** *Control* button in the Simulation Controller window, the simulation will automatically be suspended after the given number of steps are performed and you can inspect the changes of the model parameters.
In continuous processing mode the simulation will run until you press the **Suspend** *Control* button in the Simulation Controller window.

Copyright per DIN 34

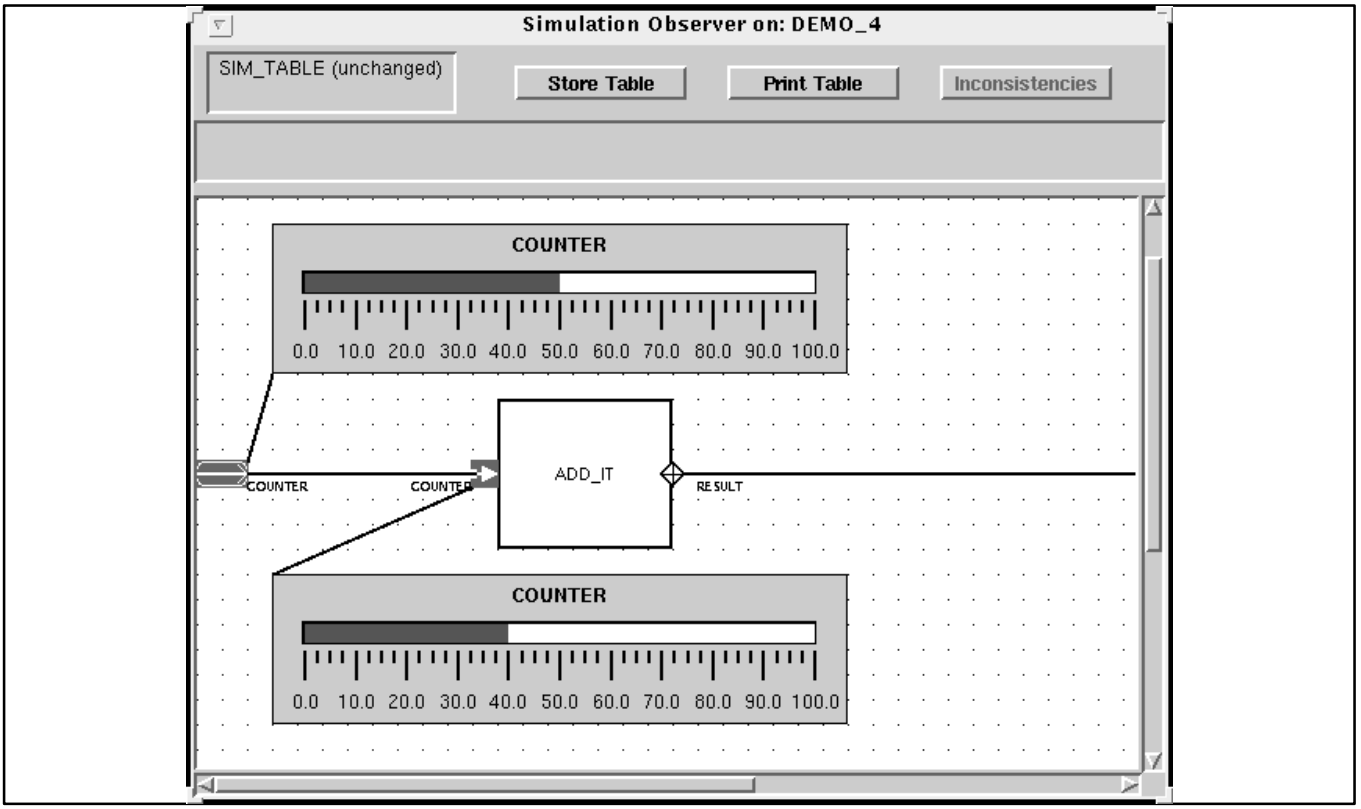


Figure 6–112 :*Monitoring the same parameter shows different values*

Figure 6–112 shows a puzzling situation. The picture shows the inside of a composite function block. The gauge element connected to COUNTER at the border of the composite displays the value 50. The monitoring element connected to the input of function block ADD_IT , COUNTER as well, shows the value 40. This situation comes up during stepwise processing mode (single step). All input buffers are written at the beginning of a time frame (this is the reason why the input connected monitoring element shows the value 40). Output buffers are written as soon as the function block has finished its calculations. (This is the reason why the other monitoring element, which is connected to an output on higher level, shows the value 50). You have to take into account such mechanisms to understand the monitoring capabilities.

6.5.6.6 The Real Time Simulation Mode

The **real time** mode will be automatically set as soon as the CSS is used in a configuration with hardware in the loop and started from VICOS. The SMT will be provided by the Master Timing Unit for all connected workstations.
The commanding is done via VICOS HCI which means that most of the CSS functions are disabled. Refer to section 6.5.7 for more information.

6.5.6.7 Creating new Simulation States

An initial simulation state was created during simulator kernel configuration. This simulation state sets the model to the start conditions.

Copyright per DIN 34

Creating a new simulation state

- Take all actions to drive the model in a state you want to use as starting point for the next simulation session.
- Move the mouse pointer into the lower part of the Simulation Controller window (simulation subview) and press the **Store** *State* button. An input window pops up.
- Type a new name in the input field.
- Press the **Apply** button.

The simulation state defines the start–up state of your simulation. If you are not interested in starting the model just from the initial state, use MOCS to create more simulation states. Start the simulation and proceed execution until the desired state is reached, then store the simulation state. The next time you can set up the simulation in this state by loading the simulation state.

6.5.6.8 Errors during simulation execution

If an error occurs during simulation run the model status changes to Running Faulty.

Note that the simulation will be continued with the last valid parameters.

All simulation results achieved in this situation have to be treated as invalid. It is possible to store the simulation state for later error evaluation. The simulation has to be stopped with the **Abort** *Control* button. This ensures the proper end of the model execution.

6.5.7 Commanding CSS via HLCL primary commands

The ICP (Immediate Command Processor) window serves as an interactive gateway to the CSS commanding via HLCL.

General informations about the use of HLCL can be found in chapter H-3.2.

Activating the ICP window

- Select a model in I_MDB.
- Select from the pop–up menu **Tools–>Start CSS .....**
- A message window saying: *Tool has been started in batch mode* appears. Press the **Ok** button.
- The CSS scope check window appears. Select the **MOCS** button with the left mouse button then press the **Start CSS** button.
- The Simulation Controller window pops up. Press the **Start ICP** button in the window's upper part (simulator subview, see Figure 6–98). The MOCS Console window appears and the ICP window pops up.

The six buttons in the ICP window (see Figure 6–113) have the following functions:

* the **History** button
  opens a seperate history window which displays all commands raised during the current session
* the **Interrupt** button
  stops the execution of a command sequence.

Copyright per DIN 34

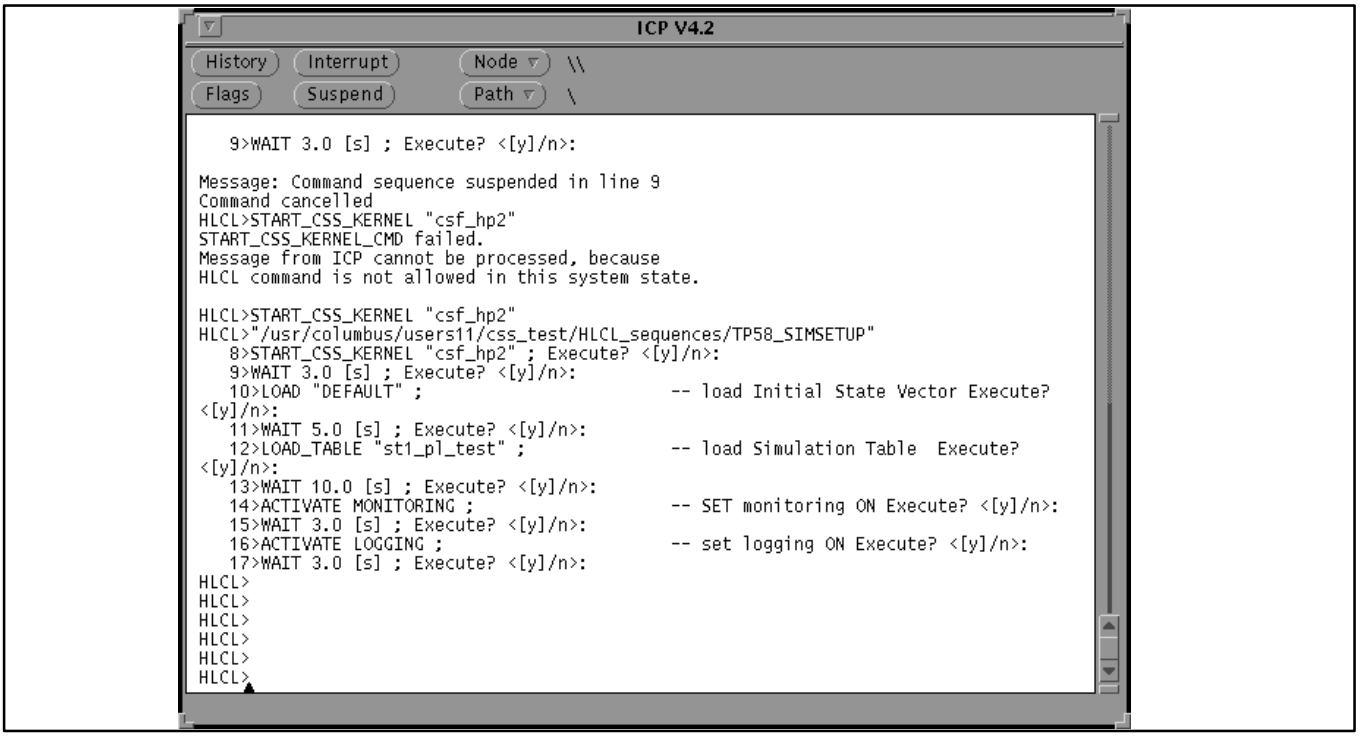


Figure 6–113 :*The ICP window is used for command sequence testing*

* the **Suspend** button
  suspends the execution of a command sequence
* the **Node** button
  allows a to change the default node. The *default node* is used to start APs and call library routines.
* the **Path** button
  allows a to change the default path. Wherever a path name is prefixed with \\, the prefix is expanded with the current value of the variable DEFAULT_PATH.
* the **Flags** button
  displays an interactive window (see Figure 6–114) and allows the setting of following predefined variables.

  ECHO:
  This variable defines whether commands executed from command sequences are echoed on the screen. If set to false, only start and stop of command sequences are reported. This variable is initially set to false.

  TRAP:
  This variable determines the error handling mechanism within command sequences. When set to true, an error interrupts the execution of the sequence and returns control to the session. The com mand sequence remains loaded and may be resumed with the RESUME command.

  STEP:
  This variable selects *single–step execution* of command sequences. When set to true, each command from a command sequence must be confirmed before execution.

Copyright per DIN 34

KEEP:
When KEEP is set to true (and a log file is open), all interactively entered commands are logged in the currently open log file.

DEBUG:
This variable selects *debug mode*. When set to true (and a log file is open), all interactively typed commands are logged in expanded form in a commands log file.

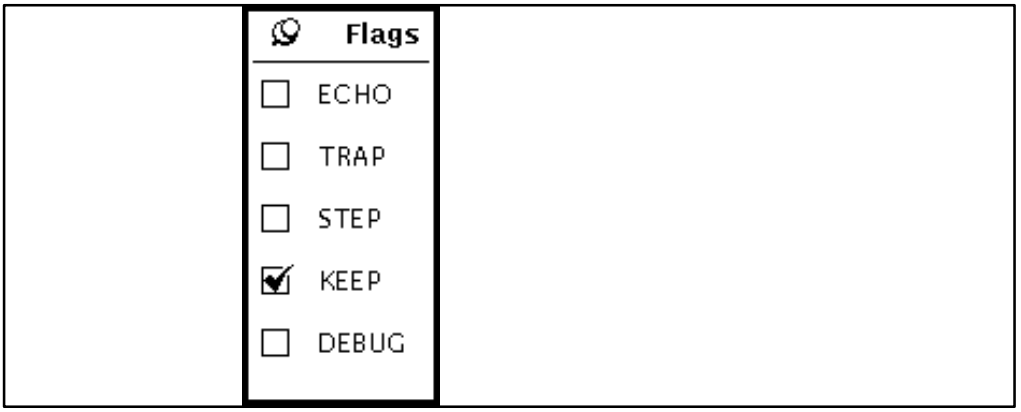


Figure 6–114 :*The flags can be set interactively*

?

On–line HLCL commanding

- An ICP window is opened, either fromthe CSS side or from the VICOS side.
- Press the **Flags** button and set the desired flags.
- Type the command and press the **Return** key on the keyboard.

Starting a HLCL command sequence

A command sequence is normally stored in the database. It is nevertheless possible that a command sequence is stored in the users own UNIX directory.

- Press the **Flags** button and select the flags ECHO, TRAP.

Note that if you select the STEP flag the user must acknowledge the execution of each step manually. Only for test purposes recommanded.

- Enter the command sequence with its pathname and press the **Return** key on the keyboard.

6.5.8 Simulation results evaluation

After the simulation is finished and the kernel is stopped, the simulation results (i.e. logging and tracing tables) are available for further investigations in the Unix file system in a directory given by the environment variable CSS_LOG_DIR.
The results are stored in ASCII files under following names:
modelname_...._username_date_time.LOG_TXT and
modelname_...._username_date_time.ARCH_TXT

Note that files with the extension .LOG and .ARCH are incomplete binary files which result from a model execution abort. These files should be deleted.

Copyright per DIN 34

In the .....ARCH_TXT you will find the function outputs & inputs logged and all external model inputs & outputs logged.

```
-----------------------------------------------------
Log value:
Runtime ID: 7  SIGNED_INTEGER, Value:         93
Log value:
Runtime ID: 9  REAL, Value:-8.88000E+01
Log value:
Runtime ID: 12  REAL, Value:-8.45774E+01
 Local Time:             15/12/1995 13:36:34.652
 Simulated Mission Time: 15/12/1995 13:35:42.118
-----------------------------------------------------
Log value:
Runtime ID: 7  SIGNED_INTEGER, Value:         93
Log value:
Runtime ID: 9  REAL, Value:-8.88000E+01
Log value:
Runtime ID: 12  REAL, Value:-8.25184E+01
 Local Time:             15/12/1995 13:36:34.852
 Simulated Mission Time: 15/12/1995 13:35:42.318
-----------------------------------------------------
```

Figure 6–115 :*A part of the file .....ARCH_TXT*

The .....LOG_TXT file contains all system messages logged, all operator commands logged and all command response logged. Additionally the file lists the runtime IDs and the related model item paths.

The file ....LOG_TXT contains the list of runtime IDS which are indispensable to read the contents of the file ....ARCH_TXT.

```
*******************************************************************************
*                                                                             *
*                              CSS LOGGER BACKEND                             *
*                                                                             *
*                                   Version:                                  *
*                                                                             *
*                      CGS/CSS Version 4.2 from 01.12.95                      *
*                                                                             *
*                                                                             *
*                 (C) 1995 DASA -- ERNO Raumfahrttechnik GmbH                 *
*                                                                             *
*******************************************************************************

Model Identification (based on Database):
  Element Configuration:´EURECA´
  Mission Name         :´DUMMY_MISSION´
  System Tree Version  : 4
  Library              :´\EURECA\SIMULATOR\TEST\MODELS_1´
  Version  : 1  Issue    : 0  Revision : 0
  Top Level FB Name    :´PL_TEST´
-----------------------------------------------------
Mapping (Runtime ID - Pathname):
 14   PL_TEST.PAYLOAD_ON
 15   PL_TEST.PAYLOAD_OFF
 7    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ERROR
 8    PL_TEST\PAYLOAD_NO_1\PL_MAX_TEMP.PL_MAX_TEMP
 9    PL_TEST\ENV_TEMP.VALUE
 10   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ACT_MFR_HEAT_CTR
 11   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.ACT_MFR_RUDO_CTR
 12   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PAYLOAD_TEMP
 13   PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_ON_OFF_STATE
 16   PL_TEST.PAYLOAD_TEMP
 17   PL_TEST.PL_ON_OFF_STATE
 3    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_MAX_TEMP
 4    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.T_ENV
 5    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_OFF_CMD
 6    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL.PL_ON_CMD
 1    PL_TEST\PAYLOAD_NO_1\PL_MAX_TEMP
 2    PL_TEST\PAYLOAD_NO_1\PAYLOAD_IMPL
-----------------------------------------------------
Command: LOGIN
```

Figure 6–116 :*The file ...LOG_TXT contains a list of runtime IDs*

Copyright per DIN 34

6.6 CGS and Application Domain Specific End Items

6.6.1 General

CGS provides a number of predefined end item types. These end items types may be complemented by application domain specific end items and data models.

End item types and their relation to aggregate types, attributes and MDA values are specified in the MDB ICD (see reference document 2.3.2). A short summary on their structure and usage is given below.

Some end item types are edited by used specific editing tools, such as the GWDU and CLS Editor/Compiler. The remaining end item types are to be edited via the I_MDB and DDED tools or can be imported by a dedicated data import tool (e.g. BDE tool).

6.6.2 End Item Concept and Usage

The end items data model concept of CGS is used to describe various data models in ground check–out, mission preparation and mission support application domains.

These data model concepts comprise:

- Data measurements
- Hardware stimuli
- Data acquisition, calibration and monitoring
- Data visualisation and display
- Telemetry and telecontrol data packet definitions
- Test environment description
- Command Languages (CPL, UCL, HLCL, dedicated libraries)
- Other data for supporting special operational concepts

Data and hardware stimuli of a unit under test or a unit under control can be accessed by a set of CGS command languages. Telecommand processing and telemetry reception can be performed by a dedicated CGS command language and dedicated CGS–SAS communication and SAS processing.

While the CPL command language is used for onboard application tasks, UCL is used both for onboard and ground application tasks. HLCL is used only for ground application tasks and telecommand processing.

All application domain specific data types have to be mapped to CGS end items and related data type definitions. Different application domain end item type definitions are available.

6.6.3 Semi–Formal Description of End Items, Aggregates and Attributes

End item types are leave types in a MDB name tree. Each MDB end item type shall be covered by a CDU (virtual) node on the path to the end item type located in the user part of the related MDB tree.

MDB name tree concepts with their related configuration control concepts such as CDU, etc are described in a dedicated section.

Copyright per DIN 34

End item types may be parameterized or non–parameterized.

A single end item consists of one or more aggregates.

Aggegates may be simple or composite.

Composite aggregates consists of an optional discriminat and a non–empty list of simple aggregates.

Simple aggregates may be single record type or multi–record type.

A simple aggregate consists of one or more attributes.

Aggregates and attributes may be optional or mandatory.

An attribute has an MDA basic data type.

MDA basic types are mapped to ORACLE data types.

- *For more detailed information see the MDB ICD.*

6.6.3.1 End Item Editing via Interactive Tools

Interactive form–oriented end item editing is supported by the tools I_MDB and DDED.

When starting the I_MDB tool (Task Selector –> Mission or Test Preparation), navigation in the MDB is provided. When navigated to an end item, values for its aggregates can be entered using the ”Open” menu entry.

An alternative way to enter data interactively is by starting the Detailed Data Editor (DDED) for an item from within the I_MDB tool.

- *Refer to MDA User Manuals for detailed description.*

6.6.3.2 End Item Creation via Batch Processing

End items may be created by batch processing or an MS–Excel type editing. This is supported by the tools BDE and EXCEL–MDB.

- *Refer to MDA User Manuals for detailed description.*

6.6.3.3 Special Tools for End Item Editing

For parameterized end items the CLS editor is needed to define the parameter types in UCL syntax.

Synoptic display end items are edited with the GWDU tool.

- *Refer to CLS and GWDU sections of this manual*

6.6.4 Basic End Item Type Definitions

6.6.4.1 Basic MDB Types and Enumerations

The description of the masks and the detailed description of the data to be entered into the mask is not provided. However, Tabel 6–5 provides information about the value ranges of the MDB basic data types and their mapping to ORACLE data types.

Copyright per DIN 34

BASIC TYPE	VALUE RANGES	ORACLE DATA TYPE
MDA BASIC DATA TYPES		
SINGLEFLOAT	–3.40282E+38 to –1.17550E–38 (negative values) 1.17550E–38 to 3.40282E38 (positive values)	VARCHAR2(22)
DOUBLEFLOAT	–1.79769313486231E+308 to –2.22507385850721E–308 (negative values) 2.22507385850721E–308 to 1.79769313486231E308 (positive values)	VARCHAR2(22)
INTEGER	– (2**31) to 2**31 –1	NUMBER
UNSIGNEDINTEGER	0 to 2**32 –1	
BITSET	32 bits long, 1 or 0	VARCHAR2(32)
HEXADECIMAL	512 bytes long, characters allowed ’0’ .. ’9’, ’A’ .. ’F’	VARCHAR2(512)
PATHNAME	Syntax of an *MDB pathname and node name :* pathname ::= \node_name {\node_name } | \node_name ::= letter_or_digit_or_underline { letter_or_digit_or_underline } *with length of node_name <= 16* letter_or_digit_or_underline ::= letter | digit | underline In the MDB the node names are stored in capital letter.	VARCHAR2(255)
STRING	– 0..MAX_STRING long, with MAX_STRING = 255.	VARCHAR2(size)
LONGCHAR	– 0..MAX_STRING long, with MAX_STRING = 32000.	LONG
RAW	– 0..MAX_RAW long, with MAX_RAW = 255.	RAW(size)
LONGRAW	– 0..MAX_RAW long, with MAX_RAW = 32000.	LONG RAW

Copyright per DIN 34

DATE	DDMMYYHH24MISS	DATE
ENUMERATION	CGS related enumerations are defined in the following table.	STRING(size)

Table 6–5 : *Basic MDA Data Types, Value Ranges and Mapping to ORACLE Data Types*

Remark:

PATHNAMES are used to designate or refere to an MDB item (any node in a MDB name tree). An attribute of type PATHNAME may be constructed by one ore more MDB item types in parantheses, as in: PATHNAME(EGSE_INTEGER_MEASUREMENT).

Predefined CGS/MDB enumerations are shown in the following Table 6–6.

CGS ENUMERATIONS	**VALUES/STRING**
CURVE_TYPE	POINT_PAIRS, LINEAR_POLYNOM, POLYNOM, IDENTICAL
BOOLEAN	TRUE, FALSE
CGS_NODE_PREFIX	HCI, CSS, TES, DBS
COMPARISON_OPERATOR	>, >=, <, <=, =, /=, in_range
CONDITION_TYPE	ENABLE_PROCESSING, SWITCH_LIMIT_SET, START_AP
COMPLETION_CODE	SUCCESS, FAILURE
EGSE_ADU_TYPE	STRUCTURED, UNSTRUCTURED, CCSDS PACKET, PUS_PACKET
EGSE_CCSDS_TIME_ID	NO TIME FIELD, REALTIME COMMAND, TIME TAG, UNDEFINED
EGSE_PACKET_TYPE	CCSDS_DEFAULT_PACKET, CCSDS_MEMORY_DUMP_PACKET, CCSDS_DATA_SEGMENT_PACKET, CCSDS_ESSENTIAL_HK_PACKET, CCSDS_SYSTEM_HK_PACKET, CCSDS_PAYLOAD_HK_PACKET, CCSDS_SCIENCE_PACKET, CCSDS_SSMB_ANCILLIARY_PACKET, CCSDS_ESSENTIAL_COMMAND_PACKET, CCSDS_SYSTEM_COMMAND_PACKET, CCSDS_PAYLOAD_COMMAND_PACKET, CCSDS_MEMORY_LOAD_PACKET, CCSDS_RESPONSE_PACKET, CCSDS_REPORT_PACKET, CCSDS_EXCEPTION_PACKET, CCSDS_ACKNOWLEDGE_PACKET
EGSE_TC_TYPE	CCSDS_TC_PACKET, PUS_TC
EGSE_BYTESTREAM_LAY-OUT_FORMAT	UNSIGNED_INTEGER, INTEGER, SINGLE_FLOAT, DOUBLE_FLOAT BINARY
EGSE_BITSTREAM_LAY-OUT_FORMAT	INTEGER, FLOAT, BINARY,UNSIGNED
EGSE_VALUE_TYPE	ASCII, HEX
EGSE_RAW_VALUE_TYPE	UNSIGNED_INTEGER, SIGNED_INTEGER, FLOAT, BYTESTREAM
EGSE_NODE_TYPE	WORKSTATION, TEST_NODE, SIMULATOR, DATABASE SERVER, FRONT–END EQUIPMENT, UNIT UNDER TEST
EGSE_SOFTWARE_TYPE	SAS —SAS executable EXECUTABLE—any FEE or UUT executable DATA_FILE —any SAS, FEE or UUT data file
EGSE_EXECUTION_MODE	NORMAL, SIMULATION, REPLAY

Copyright per DIN 34

ENTITY_STATES	DEVELOPMENT,CM_CONTROLLED
EQUAL_OPERATOR	=, /=
EQUIPMENT_TYPES	UNDEFINED, MILBUS
MIL_BUS_SLOT_CLASS	UNDEFINED, BITS_32, BITS_16, BITS_8, BITS_1, PACKET, NON_STANDARD
PACKET_PART	HEADER,SECONDARY_HEADER,DATA,CHECKSUM
PARAMETER_MODE	IN, OUT, IN_OUT
PROGRAMMING_LANGUAGES	UNDEFINED, ADA, C, ASSEMBLER, OTHER,MIXED

SW_CRITICALITIES	CAT_A, CAT_B, CAT_C, CAT_D
SW_ENTITY_TYPES	UNDEFINED, SWRU, SWEU
SW_TYPE	BOOLEAN_TYPE, STRING_TYPE, INTEGER_TYPE, UNSIGNED_INTEGER_TYPE, STATE_CODE_TYPE, BITSET_TYPE, CHARACTER_TYPE, WORD_TYPE, PATHNAME_TYPE, TIME_TYPE, COMPLETION_CODE_TYPE, SUBITEM_PATHNAME_TYPE, PULSE_TYPE, BURST_PULSE_TYPE, REAL_TYPE, LONG_REAL_TYPE
SWRU_TYPE	UNDEFINED, OFFLINE, ONLINE, EMBEDDED
WDU_SYMBOL_TYPE	STATIC, DYNAMIC
WDU_ACTION_TYPE	MENU, HLCL_COMMAND, SYNOPTIC_DISPLAY_REPLACEMENT

Table 6–6 : *Basic MDB Enumerations and Values*

Copyright per DIN 34

6.6.4.2 MDB Domain and Application Domain End Item Association

To each CDU Version a CDU MDB domain is attached specifying the valid end item classes of which end items may be defined within the CDU Version. The CDU domain can be different in several Versions of a CDU.

The MDB domain concept is a mechanism which allows a type based partition of MDB data. The domain assignment serves as a constraint restricting the different types of data available in a CDU.

Application domain end item types may be mapped to CGS end item types.

In the following sections some application specific end item domains available in CGS together with the related end items are listed.

6.6.4.2.1 Domain Specific End Items in CGS

6.6.4.2.1.1 End Items in the Domain CGS

- APID

- BOOLEAN_MEASUREMENT *
- BOOLEAN_STIMULUS *
- BOOLEAN_SW_VARIABLE *
- BURST_PULSE_STIMULUS *

*) *It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version! For TES (execution node) use DISCRETE types instead)

- CCSDS_ADU_DESCRIPTION
- CCSDS_END_POINT
- CPL_SCRIPT

- DOUBLE_FLOAT_MEASUREMENT
- DOUBLE_FLOAT_STIMULUS
- DOUBLE_FLOAT_SW_VARIABLE
- DOUBLE_FLOAT_DERIVED_VALUE

- EGSE_ANALOG_STIMULUS
- EGSE_BINARY_PACKET
- EGSE_STRING_DERIVED_VALUE

Copyright per DIN 34

- EGSE_BYTE_STREAM_MEASUREMENT
- EGSE_BYTE_STREAM_SW_VARIABLE
- EGSE_DISCRETE_DERIVED_VALUE
- EGSE_DISCRETE_MEASUREMENT
- EGSE_DISCRETE_STIMULUS
- EGSE_DISCRETE_SW_VARIABLE
- EGSE_FLOAT_DERIVED_VALUE
- EGSE_FLOAT_MEASUREMENT
- EGSE_FLOAT_SW_VARIABLE
- EGSE_INTEGER_DERIVED_VALUE
- EGSE_INTEGER_MEASUREMENT
- EGSE_INTEGER_SW_VARIABLE
- EGSE_MONITOR_LIST
- EGSE_NODE
- EGSE_PREDEFINED_TC
- EGSE_SOFTWARE
- EGSE_TEST_CONFIGURATION
- EGSE_USER_MESSAGE

- EXTERNAL_FILE

- FWDU_COMPOSITE_BINARY
- FWDU_CONVERSION_TEXT
- FWDU_DATA_DEF_RECORD_TEXT
- FWDU_DYNAMIC_OBJECT_TABLE_TEXT
- FWDU_EQUIPMENT_CONSTRAINTS
- FWDU_GIF_BINARY
- FWDU_HELP_TEXT
- FWDU_LIBRARY_BINARY
- FWDU_LIST_TEXT
- FWDU_SYMBOL_BITMAP_BINARY
- FWDU_SYMBOL_TABLE_TEXT
- FWDU_SYNOPTIC_DISPLAY
- FWDU_XWD_BINARY
- FWDU_X_BITMAP_BINARY

Copyright per DIN 34

- GDU_DESCRIPTION_LIST
- GENERAL_PURPOSES

- HLCL_COMMAND_SEQUENCE

- INTEGER_CONSTANT
- INTEGER_STIMULUS
- MEASUREMENT_LIST_TABLE
- MX_MODEL

- PULSE_STIMULUS
- PUS_ADU_DESCRIPTION
- PUS_TC
- PUS_STRUCTURE_ID
- REAL_CONSTANT
- REFERENCE_FB
- RESPONSE_PACKET

- SIMULATED_ADU_DESCRIPTION
- STRING_CONSTANT
- STRUCTURED_ADU_DESCRIPTION
- SWEU
- SWOP_COMMAND
- SWRU

- UCL_AUTOMATED_PROCEDURE
- UCL_SYSTEM_LIBRARY
- UCL_USER_LIBRARY
- UNSIGNED_INTEGER_MEASUREMENT
- UNSIGNED_INTEGER_STIMULUS
- UNSIGNED_INT_DERIVED_VALUE
- UNSIGNED_INTEGER_SW_VARIABLE
- UNSTRUCTURED_ADU_DESCRIPTION

Copyright per DIN 34

- VIRTUAL

- WDU_GROUND_SYMBOL
- WDU_GROUND_SYNOPTIC_DISPLAY

6.6.4.2.1.2 End Items in the Domain CSS

Note that the end item type TOPLEVEL_COMPOSITE_FB *can be created by the user, all other end items are created automatically during model saving.*

- ASYNCHRONOUS_FB
- COMPOSITE_FB
- CONSTANT_FB
- SYNCHRONOUS_FB
- TOPLEVEL_COMPOSITE_FB

6.6.4.2.1.3 End Items in the Domain CSS_ONBOARD

Remark: Not used in current version!

- BOOLEAN_MEASUREMENT
- BOOLEAN_STIMULUS
- BOOLEAN_SW_VARIABLE
- BURST_PULSE_STIMULUS

- DEMO_ANALOG_STIMULUS
- DEMO_BOOLEAN_MEASUREMENT
- DEMO_BOOLEAN_SW_VARIABLE
- DEMO_BYTE_STREAM_MEASUREMENT
- DEMO_DISCRETE_MEASUREMENT
- DEMO_DISCRETE_STIMULUS
- DEMO_DISCRETE_SW_VARIABLE
- DEMO_D_FLOAT_MEASUREMENT
- DEMO_D_FLOAT_SW_VARIABLE

- DEMO_FLOAT_MEASUREMENT
- DEMO_FLOAT_SW_VARIABLE
- DEMO_INTEGER_MEASUREMENT
- DEMO_INTEGER_SW_VARIABLE

Copyright per DIN 34

- DEMO_TM_PACKET
- DEMO_U_INT_MEASUREMENT
- DEMO_U_INT_SW_VARIABLE

- DOUBLE_FLOAT_MEASUREMENT
- DOUBLE_FLOAT_STIMULUS
- DOUBLE_FLOAT_SW_VARIABLE

- INTEGER_STIMULUS
- PULSE_STIMULUS

- UNSIGNED_INTEGER_MEASUREMENT
- UNSIGNED_INTEGER_STIMULUS
- UNSIGNED_INTEGER_SW_VARIABLE

- VIRTUAL

6.6.4.2.1.4 End Items in the Domain EGSE

- APID

- CCSDS_ADU_DESCRIPTION
- CCSDS_END_POINT

- DOUBLE_FLOAT_MEASUREMENT
- DOUBLE_FLOAT_SW_VARIABLE
- DOUBLE_FLOAT_DERIVED_VALUE

- EGSE_ANALOG_STIMULUS
- EGSE_BINARY_PACKET
- EGSE_STRING_DERIVED_VALUE
- EGSE_BYTE_STREAM_MEASUREMENT
- EGSE_BYTE_STREAM_SW_VARIABLE
- EGSE_DISCRETE_MEASUREMENT
- EGSE_DISCRETE_STIMULUS
- EGSE_DISCRETE_DERIVED_VALUE

Copyright per DIN 34

- EGSE_DISCRETE_MEASUREMENT
- EGSE_DISCRETE_STIMULUS
- EGSE_DISCRETE_SW_VARIABLE
- EGSE_FLOAT_DERIVED_VALUE
- EGSE_FLOAT_MEASUREMENT
- EGSE_FLOAT_SW_VARIABLE
- EGSE_INTEGER_DERIVED_VALUE
- EGSE_INTEGER_MEASUREMENT
- EGSE_MONITOR_LIST
- EGSE_NODE
- EGSE_PREDEFINED_TC
- EGSE_SOFTWARE
- EGSE_TEST_CONFIGURATION
- EGSE_USER_MESSAGE

- GDU_DESCRIPTION_LIST
- HLCL_COMMAND_SEQUENCE

- PUS_ADU_DESCRIPTION
- PUS_STRUCTURE_ID
- PUS_TC
- RESPONSE_PACKET

- SIMULATED_ADU_DESCRIPTION
- STRUCTURED_ADU_DESCRIPTION
- STRUCTURE_ID
- SWEU
- SWOP_COMMAND

- UCL_AUTOMATED_PROCEDURE
- UCL_USER_LIBRARY
- UNSIGNED_INTEGER_MEASUREMENT
- UNSIGNED_INT_DERIVED_VALUE
- UNSIGNED_INTEGER_SW_VARIABLE
- UNSTRUCTURED_ADU_DESCRIPTION

Copyright per DIN 34

- VIRTUAL
- WDU_GROUND_SYNOPTIC_DISPLAY

6.6.4.2.1.5 End Items in the Domain SDDF

Remark: Not used in current version!No

- APID

- CCSDS_END_POINT
- CPL_SCRIPT

- FWDU_COMPOSITE_BINARY
- FWDU_CONVERSION_TEXT
- FWDU_DATA_DEF_RECORD_TEXT
- FWDU_DYNAMIC_OBJECT_TABLE_TEXT
- FWDU_EQUIPMENT_CONSTRAINTS
- FWDU_GIF_BINARY
- FWDU_HELP_TEXT
- FWDU_LIBRARY_BINARY
- FWDU_LIST_TEXT
- FWDU_SYMBOL_BITMAP_BINARY
- FWDU_SYMBOL_TABLE_TEXT
- FWDU_SYNOPTIC_DISPLAY
- FWDU_XWD_BINARY
- FWDU_X_BITMAP_BINARY

- INTEGER_CONSTANT
- MX_MODEL
- REAL_CONSTANT
- RESPONSE_PACKET

- STRING_CONSTANT
- SWEU
- SWOP_COMMAND

Copyright per DIN 34

- SWRU

- VIRTUAL

6.6.4.2.1.6 End Items in the Domain UCL_LIBRARY

- INTEGER_CONSTANT
- REAL_CONSTANT
- STRING_CONSTANT
- SYSTEM_COMMAND_CHECK

- UCL_AUTOMATED_PROCEDURE
- UCL_SYSTEM_LIBRARY
- UCL_USER_LIBRARY

- VIRTUAL

6.6.4.2.1.7 End Items in the Domain DEMO_ON_BORD

- DEMO_ANALOG_STIMULUS
- DEMO_AP
- DEMO_BOOLEAN_MEASUREMENT
- DEMO_BOOLEAN_SW_VARIABLE
- DEMO_BYTE_STREAM_MEASUREMENT
- DEMO_DISCRETE_MEASUREMENT
- DEMO_DISCRETE_STIMULUS
- DEMO_DISCRETE_SW_VARIABLE
- DEMO_D_FLOAT_MEASUREMENT
- DEMO_D_FLOAT_SW_VARIABLE
- DEMO_FLOAT_MEASUREMENT
- DEMO_FLOAT_SW_VARIABLE
- DEMO_INTEGER_MEASUREMENT
- DEMO_INTEGER_SW_VARIABLE
- DEMO_TC
- DEMO_TM_PACKET
- DEMO_USER_DEFINED_1
- DEMO_USER_DEFINED_2

Copyright per DIN 34

- DEMO_USER_DEFINED_3
- DEMO_U_INT_MEASUREMENT
- DEMO_U_INT_SW_VARIABLE
- VIRTUAL

Note: See web link for actual items.

Copyright per DIN 34

6.6.5 End Item Type Descriptions

This section describes the most important CGS end items and their usage to be stored and maintained in the MDB.

- *The complete list of end item types and their aggregate decomposition can be found in the MDB ICD.*

6.6.5.1 Aggregates Common to all End Items

Aggregates	Standard Value / Default (*)	Comment
NICKNAME (single–record type)	STRING (80 character) Default(*): Last Name (Leaf) of Path	Items must be unique within CCU Nicknames must start with a character. Digits, Character and Underscore are allowed. Underscores must not be doubled and must not appear at the end. Nicknames must not be identical to UCL/HLCL keywords in case they are to be used in UCL/HLCL Nicknames are case–insensitive
| ASSIGNED PRIVILEGES (multi–record type)	STRING (80 character) Defaults are application specific	Assigned privileges must be consistent with the predefined privilege data file. Privileges are case–insensitive

6.6.5.2 Description of End Item Groups

A HTML report on at least all CGS domain end item types can be found in the CGS file location:

- *CGS_HOME/doc*

CGS configurations wich includes the Data Dictionary Maintenance Application Tool (DADIMA) can generate their own MDB data structure report in HTML format covering end item types.

- *Note: The DADIMA tool is only available on the Solaris platform!*

Generation of MDB End Item Type Report
% cd $MDA_HOME/util/dadi/bin
%start_dadima
Select “File” –> “Select Version”
The “Version” window pops up
Choose the MDB version and click on “OK” button
Select “File” –> “Print”
Click on “Data Structure Report” in the “Selected Reports” area
Click on “Exec Report” and wait
A new directory is created in the directory $MDA_HOME/util/dadi/bin/ds_report
The name of this directory is the same as the MDB version.
An internet browser must be used to read the file ds_report/<MDB–version_name>/index.htm

Copyright per DIN 34

A subset of CGS domain end item types is described below.

6.6.5.2.1 Measurements

In the checkout and test system environment, the measurement enditems describe values acquired from the unit under test, front–end equipment or a SAS via the generic input data packet ADU (see below for a description of the ADU mechanism) from a SAS. This includes telemetry data received via CCSDS packets or other telemetry formats.

Several types of measurements and corresponding MDB enditems are available, being

- EGSE_INTEGER_MEASUREMENT
- EGSE_FLOAT_MEASUREMENT
- EGSE_DISCRETE_MEASUREMENT
- EGSE_BYTESTREAM_MEASUREMENT
- UNSIGNED_INTEGER_MEASUREMENT
- DOUBLE_FLOAT_MEASUREMENT
- BOOLEAN_MEASUREMENT *

*) *It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version! For TES (execution node) use DISCRETE types instead)

The different types of measurements are defined according to the data type of the engineering value associated with this measurement, i.e. a pressure described as a database enditem will have a floating point value associated with it (e.g. 5.6 bar) whereas a counter will have an integer value. Discrete measurements have state codes as engineering values and data in string format (i.e. ASCII strings) are defined in bytestream measurements.

Enditems declared as measurements can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname.

The overall properties of all measurement enditems in MDB are the same and comprises:

- a ***PUS Parameter description*** which is **optional**. This description describes the type of the TM downlink parameter as used in the onboard systems
- a ***raw value description*** which is **mandatory**. This description describes the type of the raw value of a measurement, e.g. data type, size, as acquired in the ADU.
- an ***engineering value description*** which is **mandatory** for the integer and float measurements but not required for the discrete and bytestream ones. This description describes the type of the engineering value of a measurement, e.g. value ranges.
- a ***calibration description*** which is **mandatory**. This description describes the transformation of the described raw value into the (described) engineering value during run–time, e.g. a calibration of an integer value via a polynom into a floating point value.
- an ***engineering value logging definition*** which is **optiona**l. This definition allows to define in MDB whether the value shall be logged whenever being updated later during run–time.
- a ***physical address***, which is **optional**. This information is needed by the SAS to physically acquire the value and is not processed by CGS. This information is optional (for an individ-

Copyright per DIN 34

ual measurement) since a SAS may know how to acquire the data or may use global physical address information stored with the ADU description.

- a ***monitoring definition*** which is **optional**. This property defines the limits against which the actual value of the measurement shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)
- a condition ***definition*** which is **optional**. This property defines the conditions which become active, if the measurement's value fulfills the specified comparison and the associated action.

6.6.5.2.1.1 EGSE_INTEGER_MEASUREMENT

This end item type describes a measurement acquired via the ADU service from a SAS with an *integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.1.2 UNSIGNED_INTEGER_MEASUREMENT

This end item type describes a measurement acquired via the ADU service from a SAS with an *unsigned_integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.1.3 EGSE_FLOAT_MEASUREMENT

This end item type describes a measurement acquired via the ADU service from a SAS with a *float* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.1.4 DOUBLE_FLOAT_MEASUREMENT

This end item type describes a measurement acquired via the ADU service from a SAS with a *double float* (64 bit) engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.1.5 EGSE_DISCRETE_MEASUREMENT

This end item type describes a measurement acquired,e.g. via ADU service from a SAS with a *discrete* engineering value. This end item is also the counterpart to the CSS top level I/O of type state code.
The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system

Copyright per DIN 34

will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

This End Item is also used for acquisition of boolean values of a 1-Bit digital measurement device from SAS interfacing to frontends or measurement devices, or sent within telemetry packets (i.e. all values acquired via SAS).

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.1.6 EGSE_BYTESTREAM_MEASUREMENT

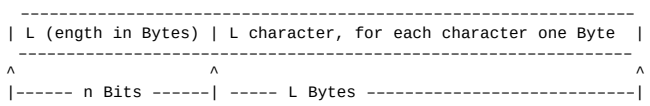
This end item type describes a measurement acquired via the ADU service from an SAS with a *bytestream* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

An EGSE_BYTESTREAM_MEASUREMENT can be calibrated from raw value type INTEGER/UNSIGNED_INTEGER by defined mapping table or calibrated from raw value type BYTESTREAM.

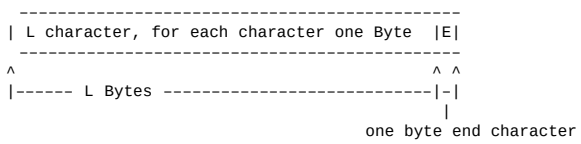
An EGSE_BYTESTREAM_MEASUREMENT calibrated from raw value type BYTESTREAM can be defined as STATIC with fixed length or as DYNAMIC with variable length. For dynamic EGSE_BYTESTREAM_MEASUREMENTs a leading length field (see A) or an end character (in case of missing length field or length field = 0, see B) must be defined.

Alternative A (string with previous length field): The string is split into two parts:



with the string characters following the length definition immediately.

Alternative B (end character terminated string): The string is split into two parts:



with the string characters terminated by end character.

In case of Dynamic String the ’Raw Value Size in Bits’ is obsolete.

6.6.5.2.1.7 BOOLEAN_MEASUREMENT

This end item type describes a measurement acquired with a *boolean* engineering value.

- *It is used by CSS/CMAS only.*
  (CSS/CMAS are not supported in current version! For TES (execution node) use DISCRETE types instead)

Copyright per DIN 34

6.6.5.2.2 SW Variables

In the checkout environment, the SW variable enditems describe values locally maintained in the Test Nodes. This includes

- nternal status data (maintained in Ada) of the products TES, the SW running on test nodes, and DBS (the central test result database)
- user defined SW variables which are then available in UCL/HLCL as place holders for writing/reading
- data. SW variables can be read (any type) or written (only the UCL SW variables not mapped to internal
- status data) from SAS also: Software Variables may be updated via the TES_API using the WRITE_VALUE services (i.e. by directly writing values to end items).

Enditems declared as SW variables can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, SAS etc. via their pathname.

Several sub–types of SW variables are available:

- EGSE_INTEGER_SW_VARIABLE
- UNSIGNED_INTEGER_SW_VARIABLE
- EGSE_FLOAT_SW_VARIABLE
- DOUBLE_FLOAT_SW_VARIABLE
- EGSE_DISCRETE_SW_VARIABLE
- EGSE_BYTESTREAM_SW_VARIABLE
- BOOLEAN_SW_VARIABLE *

*) *It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version! For TES (execution node) use DISCRETE types instead)

The different types of SW variables are defined according to the data type of the engineering value associated with it, i.e. a pressure described as a database enditem will have a floating point value associated with it (e.g. 5.6 bar) whereas a counter will have an integer value. Discrete SW variables have state codes as engineering values and data in string format (i.e. ASCII strings) are defined in bytestream SW variables.

The overall properties of all SW variable enditems in MDB is the same and comprises:

- an *initial value description* which is **mandatory**. This description defines the initial value of the SW variable. This is necessary because a value can be assigned from UCL/HLCL and until this happens for the first time the value would be undefined.
- an *engineering value description* which is **mandatory** for the integer and float SW variables but not required for the discrete and bytestream ones. This description describes the type of the engineering value of a measurement, e.g. value ranges.
- an *HK source definition* which is **optiona**l. This property defines to which housekeeping value the SW variable shall be bound to. Housekeeping values are internal status data of Ada SW (CGS Processes) A list of all available housekeeping values which can be linked with SW variables can be found in teh resp. chapter of this document.

Copyright per DIN 34

- an *engineering value logging definition* which is **optional**. This definition allows to define in MDB whether the value shall be logged whenever being updated later during run–time. (Note: This flag can be overwritten by online UCL commands)
- a *monitoring definition* which is **optional**. This property defines the limits against which the actual value of the SW variable shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)

6.6.5.2.2.1 EGSE_INTEGER_SW_VARIABLE

This end item type describes a SW variable with an *integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.2.2 UNSIGNED_INTEGER_SW_VARIABLE

This end item type describes a SW variable with an *unsigned_integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.2.3 EGSE_FLOAT_SW_VARIABLE

This end item type describes a SW_VARIABLE with a *float* engineering value.

This End Item is used for acquisition of values from software (e.g. AP, SAS), which engineering unit is a float. This is updated via the TES_API using the WRITE_VALUE services (i.e. by directly writing values to end items), or for values that are generated internally (housekeeping values).

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.2.4 DOUBLE_FLOAT_SW_VARIABLE

This end item type describes a SW variable with an *unsigned_integer* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.2.5 EGSE_DISCRETE_SW_VARIABLE

This end item type describes a SW_VARIABLE with a *state code* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an AP.

6.6.5.2.2.6 EGSE_BYTESTREAM_SW_VARIABLE

This end item type describes a SW_VARIABLE with a *bytestream* engineering value.

Copyright per DIN 34

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.2.7 BOOLEAN_SW_VARIABLE

This end item type describes a SW variable with an *boolean* engineering value.

It is used by CSS/CMAS only.
(CSS/CMAS are not supported in current version! For TES (execution node) use DISCRETE types instead)

6.6.5.2.3 Derived Values

In the checkout environment, Derived Value end items describe values calculated according to a mathematical/logical expression. A derived value can be displayed and monitored as any measurement and software variable.

The following types of Derived Values are available:

- EGSE_INTEGER_DERIVED_VALUE
- UNSIGNED_INT_DERIVED_VALUE
- EGSE_FLOAT_DERIVED_VALUE
- DOUBLE_FLOAT_DERIVED_VALUE
- EGSE_DISCRETE_DERIVED_VALUE
- EGSE_STRING_DERIVED_VALUE

The different types of Derived Values are defined according to the data type of the respective engineering value.

End items declared as Derived Values can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname.

The overall aggregate structure of all Derived Values end items in MDB is the same and comprises:

- an Engineering Value Description which is mandatory for the integer and float Derived Values, but not required for the discrete ones.
- The source code of an expression in UCL syntax specifying the calculation of the value
- The compiled items for the UCL expression, as there are
  – the I–Code
  – the symbol tables
  – the cross reference table
  – the compilation date and the compilation status
- an Engineering Value Logging flag which is optional. It indicates whether or not the value shall be logged.
- a Monitoring Definition which is optional. This aggregate defines the limits against which the actual value of the SW variable shall be checked and which actions shall be taken in case the limits are violated (out of limit situation)
- an optional Condition Definition. This aggregate defines conditions for the enditem, which control the processing or monitoring of other enditems or the starting of APs.

The source code of the expression must follow the following rules:

Copyright per DIN 34

– UCL syntax is applicable (refer to UCL LRM). The following subset of statements is allowed:
  – import statement
  – if statement
  – case statement
  – return statement
  – function calls
    (to standard functions/procedures and system libraries)
  – use of unitized values
  – type conversions and call to standard functions

– The following is not allowed:
  –no declarations
  –no assignments
  – import allowed only for system libraries, not for user libraries
  – Only the following declarations are possible:
    — Constants
    — Type declarations
    — Alias
  – no procedure calls
  – No assignments to variables or MDB Objects

Note: The restrictions are introduced to avoid overloading of the monitoring function with UCL processing and to avoid execution of any library procedure (e.g. sending a command, disable archiving etc.) from within derived value expressions.

-

Example of Expression for an Integer_Derived_Value:

```
return \satellite\ss\active_count + \satellite\boxes\active_count;
```

-

Example of Expression for an Egse_Float_Derived_Value:

```
if \satellite\powerstatus = $ON then
   return \sat\subsys_a\power\current + \sat\subsys_a\power\current;
else
   return 0.0 [A];
end if;
```

-

Example of Expression for an Egse_Discrete_Derived_Value:

```
if \satellite\powerstatus = $ON then
   return $POWER;
else
   return $UNDEF;
end if;
```

-

Example of Expression for an Egse_String_Derived_Value:

Copyright per DIN 34

```
if \satellite\powerstatus = $ON then
   return “power is active”;
else
   return “power is inactive”;
end if;
```

6.6.5.2.3.1 EGSE_INTEGER_DERIVED_VALUE

This type describes Derived Value end items with *integer* engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.3.2 UNSIGNED_INT_DERIVED_VALUE

This type describes Derived Value end items with unsigned_integer engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.3.3 EGSE_FLOAT_DERIVED_VALUE

This type describes Derived Value end items with *float* engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.3.4 DOUBLE_FLOAT_DERIVED_VALUE

This type describes Derived Value end items with *double float* engineering values.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

6.6.5.2.3.5 EGSE_DISCRETE_DERIVED_VALUE

This enditem type describes a SW_VARIABLE with a *state code* engineering value.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an AP.

6.6.5.2.3.6 EGSE_STRING_DERIVED_VALUE

This enditem type describes a Derived Value with an engineering value of type *string*.

The values can be displayed, monitored and logged. In case of limit violations, the checkout and test system will generate exception messages and initiate actions such as sending a command or starting an Automated Procedure.

Copyright per DIN 34

6.6.5.2.4 Messages

User messages are a convenient mechanism to provide more useful information to the user in case of erroneous or abnormal situations, e.g. when monitoring the incoming data from the unit under test.

EGSE_USER_MESSAGES can be refered to in monitoring definitions of measurements / SW Variables (Attribute ”Exception Message”). The message is then sent to the Message Window and to the Event Log in case of limit violations.

6.6.5.2.4.1 EGSE_USER_MESSAGE

6.6.5.2.5 Test Facility Description

There are a number of database enditems which describe the test facility and the configuration of the facility for a given test. This includes definitions of the individual HW nodes of the test equipment, the individual SW entities and the overall configuration. All items can be configured to a test configuration which can be selected during test setup and activated for a given test. This test configuration defines then the participating nodes, the needed SW entities , the CCU to be applied as well as individual items to be loaded to the test nodes.

Note: For a defined Test Configuration, the Tool “Generate_Scoe_Files” must be called within a slected CCU, whenever items allocated to the Test Configuration have been changed.

6.6.5.2.5.1 EGSE_NODE

This end item describes a specific part of the HW of the facility. Its main purpose is to provide access to parts of the facility via pathnames from UCL/HLCL level. It also defines the ’role’ of the node, e.g. in the EGSE environment it determines whether the node is a workstation, a test node, a database server, a simulation node, a front–end equipment or a UUT. Finally, configuration management of this piece of the facility can be performed using the standard MDB Configuration Management provisions and the Test Setup and Configuration (TSCV) program.

6.6.5.2.5.2 EGSE_SOFTWARE

This end item describes a specific part of the SW of the facility. Its main purpose is to provide access to this SW via pathname from UCL/HLCL level. It also defines the ’type’ of the SW, e.g. in the EGSE environment it determines whether the SW is a SAS, a data file, etc. Finally, configuration management of this piece of the facility can be performed using the standard provisions of the MDB

6.6.5.2.5.3 EGSE_TEST_CONFIGURATION

This end item describes the actual configuration of the facility for a given checkout application. It is the actual configuration of the test equipment to be set–up for a given test.

The test configuration is described in terms of references to other MDB items of type EGSE_NODE and EGSE_SW plus the definition of MDB contents (CDUs) to be ’downloaded’ to test nodes. For test nodes, the role of the Node (being the Master Test Processor or ordinary test node) , the execution mode and predefined items such as the AP to be executed when started in batch mode, or the allocated overview synoptic can be defined.

For test nodes and workstations, a flag indicates, if the node is foreseen to participate. This flag can be modified during the setup in TSCV.

Copyright per DIN 34

End items of type test configuration are later used in the CGS product TSCV to actually perform the set-up. Several Test Configurations may be defined. Up to 5 can be loaded and activated in parallel, given that their nodes do not participate in more than one of them.

6.6.5.2.6 Generation Data Units, Stimuli and Telecommands

In the checkout environment, the stimuli enditems describe commands to be sent to the unit under test, front–end equipment or a SAS via the generic output data packet GDU. GDU stands for ”Generation Data Unit” and is another word for a data block sent to SAS. The internal structure of this data block needs not to be known when filling the MDB with data. However, the purpose of the stimulus has to be known, i.e. whether it is meant for the SAS, a front end or the unit under test and how it shall be sent (via a dedicated HW interface, a front–end equipment or the ground to space link. Thus, stimuli include telecommands sent via CCSDS packets or other telecommand formats.

Several types of stimuli and corresponding MDB enditems are available, being:

- EGSE_ANALOG_STIMULUS
- EGSE_DISCRETE_STIMULUS
- EGSE_BINARY_PACKET
- EGSE_PREDEFINED_TELECOMMAND
- PUS_TC
- INTEGER_STIMULUS *
- UNSIGNED_INTEGER_STIMULUS *
- DOUBLE_FLOAT_STIMULUS *
- BOOLEAN_STIMULUS *
- PULSE_STIMULUS *
- BURST_PULSE_STIMULUS *

**) It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version!)

The different types of stimuli are defined according to the type of data associated with it, i.e. an analog stimulus comes along with an analog (float) value (e.g. a command to set the output voltage of a power supply to 28.0 Volt) a discrete stimulus has a discrete value (e.g. switch a pump ON) or a predefined TC has a bit pattern associated which is the telecommand to be sent (e.g. a CCSDS telecommand).

Stimuli are ”parameterized” items. Their type definitions comprise a formal parameter list describing the type of values that must (or may) be supplied at runtime.

Enditems declared as stimuli can be referred to in UCL APs, procedures or functions, HLCL sequences, synoptic displays, etc. via their pathname and then result in the defined command being sent to SAS.

Enditems declared as stimuli can also be referred to in monitoring definitions as actions, meaning that they are sent to the SAS when the out of limit condition is reached.

In simulation models, stimuli are used to describe input/output values of the models. The may be received from external sources (CMAS and front ends)

Copyright per DIN 34

In checkout environments (VICOS), all stimuli/telecommnds/packets are refered to as Generation Data Units (GDU).

The overall property structure of all stimuli enditems in MDB is comprises:

- a ***general descripti****on* which is **mandatory**. This description contains some identification information which is needed for CGS and the SAS to properly handle the command.
- a ***physical address****,* which is **optiona**l. This aggregate contains information needed by the SAS/CMAS to physically process the command and is not processed by CGS.
- a ***raw value*** *description* which is **optional**. This description is not needed for checkout but for simulation purposes only.
- an ***engineering value*** *description* which is **optional**. This description is not needed for checkout but for simulation purposes only.
- a *parameter list* which is **optional**. If present, 1 to 255 elements can be defined. The parameters are substituted into the GDU actually generated at run–time. Only a constraint number of data types is allowed as parameter.
- for each parameter, a ***de–calibration*** *description* which is **optiona**l.

  In the checkout environment, the decalibration allows to specify engineering units which are converted/decalibrated to raw values before they are sent / inserted into the packet/GDU. For simulation models, the stimuli are received.Therefore their values are calibrated.
- a *Command* ***Pre–Conditio****n Description,* which is **optional**

  In the checkout environment, a Pre–Condition may be verified before a stimulus/TC Packet is sent , by checking that referenced measurements/sw variables/derived values have a specified value.
- a *Command* ***Verification*** *Description,* which is **optional**

  In the checkout environment, a stimulus/TC Packet is sent and afterwards, a verification may be performed by checking that referenced measurements/sw variables/derived values have a specified value.

  The descriptions includes two times:

  Activation_Delay_in_Seconds
  A float value describing the number of seconds to wait before the verification of the measurements is started

  Verification_Timeout_in_Seconds
  A float value describing the number of seconds to wait after the ACTIVATION_DELAY has expired and before the verification of the measurements is finished
- a *Command* ***Authorization****,* which is **optional**

  The descriptions includes two items:
  The declaration, that the command is critical, and thus is to be authorized/confirmed before sending.
  and, optionally, an additional ’password’ (string) to be entered by the user in the confirmation window.

Copyright per DIN 34

6.6.5.2.6.1 EGSE_ANALOG_STIMULUS

This end item describes a command to be sent to a SAS and/or a front end equipment and/or the unit under test comprising one analog value as a parameter.

6.6.5.2.6.2 EGSE_DISCRETE_STIMULUS

This end item describes a command to be sent to a SAS and/or a front end equipment and/or the unit under test comprising one discrete value as a parameter. This end item is also the counterpart to the CSS top level I/O of type state code.

6.6.5.2.6.3 EGSE_BINARY_PACKET

This is a parameterizable MDB end–item defining a packet of data items sent within a data buffer.

This end item type defines a stimulus having a stream of binary data as contents. The enditem is sent to an SAS within a GDU. It is a *parameterized* end item type with up to 255 parameters. The data stream associated with the enditem may consist of up to 4096 bytes. Similar to the other stimulus definitions, the enditem definition contains some general information, physical address information and a parameter list. In addition, however, the necessary data to describe the layout of the binary packet has to be provided in the following aggregates:

- Bit stream layout description. This aggregate describes the predefined static bit patterns used for initializing the binary packet, i.e. the byte stream associated with it.
- a *Command Verification Description*, which is **optional**

  See EGSE_ANALOG_STIMULUS
- ***Parameter Position Role*** *Description*, which is optional
  In this description the three parameters have following meanings:
  - ’***Location Specification Mode***’ with possible values:
    - ABSOLUTE (Default) => the offset definition of a parameter is absolute to the start of the given packet part, starting with 1. In this mode the next two parameters (’Size of String Length Field’, ’Parameter Alignment’) are not significant. String parameter are handled as static (predefined) strings, dynamic strings are not allowed.
    - ALIGNED => all parameters are aligned depend on the ’Parameter Alignment’ definition, the offset definition of a parameter is not significant for this parameter. String parameter are handled as dynamic strings.
    - RELATIVE => the offset definition of a parameter denotes the offset to the end of the previous parameter. Offset 1 means the parameter follow the previous immediately. String parameter are handled as dynamic strings.
  - For ALIGNED and RELATIVE modes only you can define dynamic string parameter only. The parameter ’***Size of String Length Field***’ denotes the length of the string length field. A dynamic string contains the string length field followed by the string itself.
  - For ALIGNED mode only you can define ’***Parameter Alignment***’ for the definition of alignment.

Copyright per DIN 34

Following alignments are possible:

BIT_8_ALIGNED_MSB
BIT_16_ALIGNED_MSB
BIT_32_ALIGNED_MSB
BIT_8_ALIGNED_LSB
BIT_16_ALIGNED_LSB
BIT_32_ALIGNED_LSB
UNALIGNED

Alignmnet examples for three parameter:

```
  Alignment        |   Parameter in Packet
                   |   (Bit location starting with 1)
 ------------------|-------------------------------------------------------------
 ----------------  |
| Name | Bit Size | |
|------+----------| |
|  a   |   4      | |   +--------+--------+--------+--------+--------+--------+
|  b   |  12      | |   |        | 1      |   2    |     3  |       4|        |
|  c   |   5      | |   |12345678|90123456|78901234|56789012|34567890|12345678|
 ----------------  |   +--------+--------+--------+--------+--------+--------+
                   |   |        |        |        |        |        |        |
 BIT_8_ALIGNED_MSB |   |aaaa....|bbbbbbbb|bbbb....|ccccc...|........|........|
                   |   |        |        |        |        |        |        |
 ------------------|-  +--------+--------+--------+--------+--------+--------+
                   |   |        |        |        |        |        |        |
 BIT_8_ALIGNED_LSB |   |....aaaa|....bbbb|bbbbbbbb|...ccccc|........|........|
                   |   |        |        |        |        |        |        |
 ------------------|-  +--------+--------+--------+--------+--------+--------+
                   |   |        |        |        |        |        |        |
 BIT_16_ALIGNED_MSB|   |aaaa....|........|bbbbbbbb|bbbb....|ccccc...|........|
                   |   |        |        |        |        |        |        |
 ------------------|-  +--------+--------+--------+--------+--------+--------+
                   |   |        |        |        |        |        |        |
 BIT_16_ALIGNED_LSB|   |........|....aaaa|....bbbb|bbbbbbbb|........|...ccccc|
                   |   |        |        |        |        |        |        |
 ------------------|-  +--------+--------+--------+--------+--------+--------+
                   |   |        |        |        |        |        |        |
 UNALIGNED         |   |aaaabbbb|bbbbbbbb|ccccc...|........|........|........|
                   |   |        |        |        |        |        |        |
                   |   +--------+--------+--------+--------+--------+--------+
```

- Formal parameter list (up to 255 parameters are allowed).
  For each parameter:
  - A common parameter declaration source, defining the names, the types and the default values for the parameters in UCL syntax. This source definition must be compiled with the CLS_Editor (to be called via the “tools” menu entry of I_MDB)
  - The following parameter types are supported:
    STRING,STATECODE,INTEGER,UNSIGNED_INTEGER,REAL,LONG_REAL,PATH NAME,TIME,BYTE,WORD,LONG_WORD,BYTE_STRING

Copyright per DIN 34

- The definition of the place in the data buffer where the parameter shall be replaced within the packet during online sending: offset, and length
- a decalibration description which is optional. Depending on the parameter type, this may be analog (INTEGER, UNSIGNED_INTEGER,REAL,LONG_REAL) or discrete (STATE_CODE) decalibration
- an optional raw value description.
- an optional engineering value description.

Note: Parameter are specified by the CLS Editor as usual, e.g. :
(PARA1: INTEGER := 3.14);

The meaning of the formal parameter list defined with this enditem is somehow different from the one defined for the other stimuli enditems. Here, the formal parameters defined for the binary packet will be substituted at run–time into the data buffer by CGS. The logic with respect to the handling of the bitstream layout and parameters is that first the static bit patterns defined by bitstream layout is put into the data buffer and then the actual values of the formal parameters will be substituted, possibly overwriting the static bits.

Note: Binary Packets are issued via the UCL Issue command, as in:
ISSUE (\A\B\C (123, 1.0, 100, $ON), stat);

The end item is only used for checkout (VICOS).

6.6.5.2.6.4 EGSE_PREDEFINED_TC

This is a parameterizable MDB end item defining a complete CCSDS packet with header and data information.

This end item defines a CCSDS telecommand to be sent to the unit under test by SAS. Similar to the other stimulus definitions, it contains some general information, physical address information and a parameter list. In addition, however, the necessary data to describe the layout of the CCSDS packet has to be provided in the following properties:

- a ***CCSDS (primary) header*** *description* which is **mandatory**. This property defines the contents of all fields of the primary header of the CCSDs packet which have to be predefined before packet generation.
- a ***CCSDS secondary header*** *description* which is optional. This property defines the contents of all fields of the secondary header of the CCSDs packet which have to be predefined before packet generation.
- a ***bitstream layout*** *description* which is **mandatory**. This property describes the static bit pattern to be put into the packet data field (after the secondary header, if present or after the primary header if no secondary header available). The bitstream layout can be used to define op–codes for specific onboard commands, for example.
- a ***command buffer item description*** which is optional. This description is not needed for checkout but for simulation purposes only.
- a ***Command Verification*** *Description*, which is optional

See EGSE_ANALOG_STIMULUS

Copyright per DIN 34

- a ***Command Authorization Description***, which is optional

  See EGSE_ANALOG_STIMULUS

- ***Parameter Position Role*** *Description*, which is optional
  See EGSE_BINARY_PACKET (section 6.6.5.2.6.3 on page 6–237)

- up to 255 ***parameters,*** and for each:

  - A common parameter declaration source, defining the names, the types and the default values for the parameters in UCL syntax. This source definition must be compiled with the CLS_Editor (to be called via the “tools” menu entry of I_MDB)

  - The following parameter types are supported:
    STRING, STATECODE, INTEGER, UNSIGNED_INTEGER, REAL, LONG_REAL, PATHNAME, TIME, BYTE, WORD, LONG_WORD, BYTE_STRING

  - The definition of the place where the parameter shall be replaced within the packet during online sending: packet part (header, 2nd header, data, checksum), offset and length

  - a decalibration description which is optional. Depending on the parameter type, this may be analog (INTEGER, UNSIGNED_INTEGER,REAL,LONG_REAL) or discrete (STATE_CODE) decalibration

  - a raw value description

  - an engineering value description

The meaning of the formal parameter list defined with this enditem is somehow different from the one defined for the other stimuli enditems. Here, the formal parameters defined for the telecommand will be substituted at run–time into the data section of the CCSDS packet (see above for a definition) by CGS. The logic with respect to the handling of the bitstream layout and parameters is that first the static bit patterns defined by bitstream layout is put into the data section and then the actual values of the formal parameters will be substituted, possibly overwriting the static bits.

- *The end item is only used for checkout (VICOS).*

The general format of CCSDS packet headers is shown in the figure below.

Copyright per DIN 34

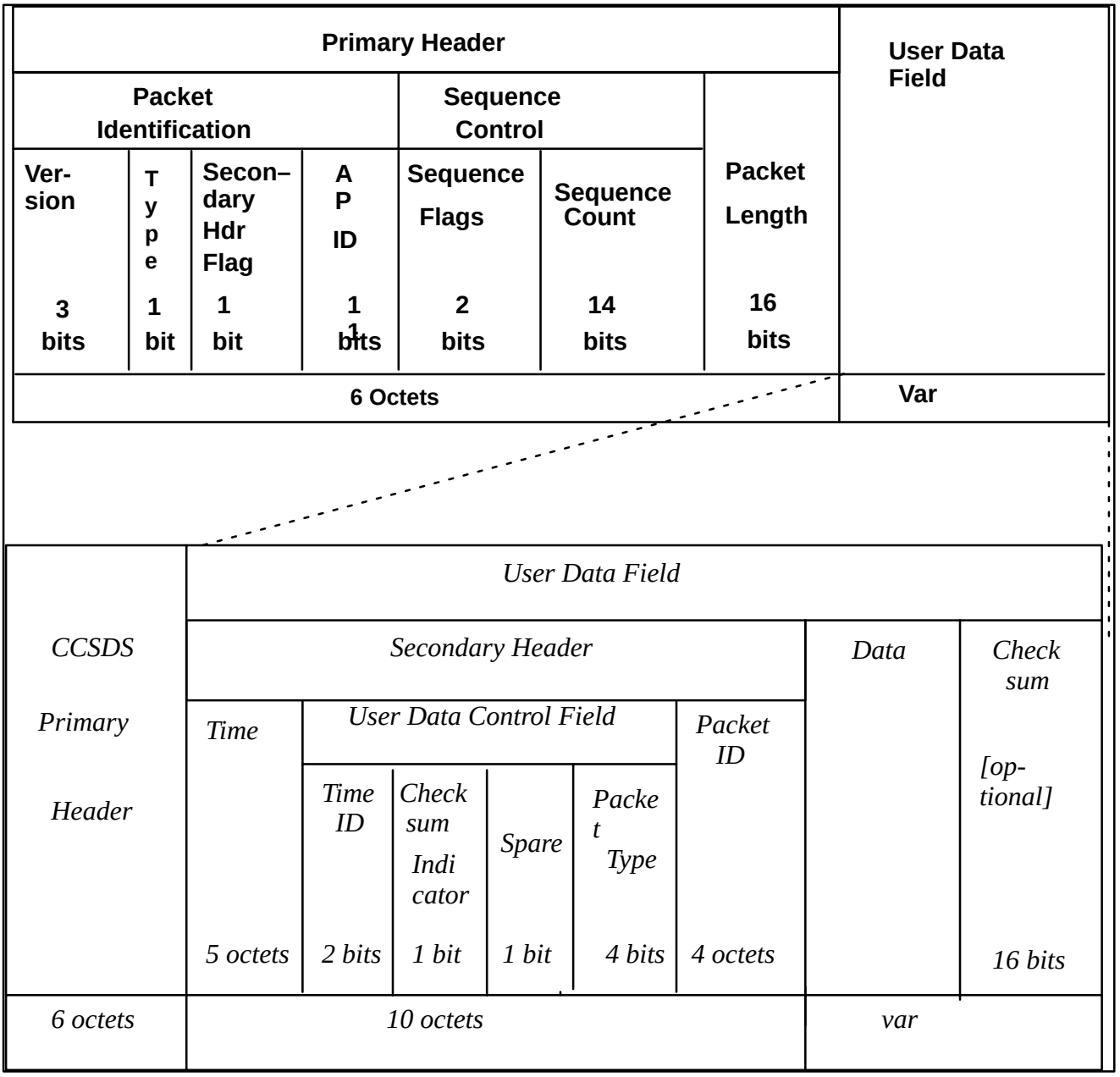


Figure 6–117 :*CCSDS Primary and Secondary Header*

6.6.5.2.6.5 PUS_TC

This is a parameterizable MDB end item defining a complete CCSDS packet according to the ESA Packet Utilisation Standard (PUS) with header and data information.

This enditem defines a CCSDS telecommand to be sent to the unit under test by SAS. It is very similar to the EGSE_PREDEFINED_TC end item, except for the secondary header aggregate.

- a ***CCSDS (primary) header*** *description* which is **mandatory**. This property defines the contents of all fields of the primary header of the CCSDs packet which have to be predefined before packet generation.

Copyright per DIN 34

- a ***CCSDS secondary header*** *description* which is optional. This property defines the contents of all fields of the datafield header for the PUS TC packet which have to be predefined before packet generation.
- a ***bitstream layout*** *description* which is **mandatory**. This property describes the static bit pattern to be put into the packet data field (after the secondary header, if present or after the primary header if no secondary header available). The bitstream layout can be used to define op–codes for specific onboard commands, for example.
- a ***command buffer item description*** which is optional. This description is not needed for checkout but for simulation purposes only.
- a ***Command Verification*** *Description*, which is optional

  See EGSE_ANALOG_STIMULUS
- a ***Command Authorization Description***, which is optional

  See EGSE_ANALOG_STIMULUS
- ***Parameter Position Role*** *Description*, which is optional
  See EGSE_BINARY_PACKET (section 6.6.5.2.6.3 on page 6–237)
- up to 255 ***parameters,*** and for each:
  - A common parameter declaration source, defining the names, the types and the default values for the parameters in UCL syntax. This source definition must be compiled with the CLS_Editor (to be called via the “tools” menu entry of I_MDB)
  - The following parameter types are supported:
    STRING, STATECODE, INTEGER, UNSIGNED_INTEGER, REAL,LONG_REAL, PATHNAME, TIME, BYTE, WORD, LONG_WORD, BYTE_STRING
  - The definition of the place where the parameter shall be replaced within the packet during online sending: packet part (header, 2nd header, data, checksum), offset and length
  - a decalibration description which is optional. Depending on the parameter type, this may be analog (INTEGER, UNSIGNED_INTEGER,REAL,LONG_REAL) or discrete (STATE_CODE) decalibration
  - a raw value description
  - an engineering value description

6.6.5.2.6.6 INTEGER_STIMULUS

This enditem is used by CSS/CMAS only.

The Integer Stimulus type is a parameterized end–item type that takes one integer value as parameter.

6.6.5.2.6.7 UNSIGNED_INTEGER_STIMULUS

6.6.5.2.6.8 DOUBLE_FLOAT_STIMULUS

The Double Float Stimulus type is a *parameterized* end–item type that takes one double–float value as parameter.

Copyright per DIN 34

It is used by CSS/CMAS only.

6.6.5.2.6.9 BOOLEAN_STIMULUS

**) It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version!)

The Boolean Stimulus type is a parameterized end–item type that takes one boolean value as parameter.

6.6.5.2.6.10 PULSE_STIMULUS

**) It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version!)

The pulse stimulus is a stimulus which creates a pulse.

6.6.5.2.6.11 BURST_PULSE_STIMULUS

**) It is used by CSS/CMAS only.*
(CSS/CMAS are not supported in current version!)

6.6.5.2.7 SWOP Commands and Response Packets

6.6.5.2.7.1 SWOP_COMMAND

This is a parameterizable MDB end item defining a specific CCSDS packet for COF/ISSA.

This end item defines a packet that is sent to the Onboard System (via a SAS) as a SWOP command. A SWOP Command is defined by its SID and the set of parameters. A SWOP command is normally answered by the onboard system with a telemetry packet (response packet).

Similar to other stimulus definitions, it contains some general information and a parameter list. No physical address information is necessary, as the sending path is defined by the Application ID , which is given as an online parameter (derived from the CCSDS End Point). The CCSDS Primary Header is created by the sending software (VICOS/TES). The following information describes a SWOP command:

- a *CCSDS secondary header description* which is **optional**. This property defines the contents of all fields of the secondary header of the CCSDs packet which have to be predefined before packet generation.
- up to 255 *parameters*
- a reference to a response packet which is to be received after the command has been sent
- a reference to the SWRU which executes the SWOP within the onboard system

The formal parameters will be substituted at run–time into the data section of the CCSDS packet (see above for a definition) by VICOS/TES according to a predefined, fixed schema. No decalibration is foreseen.

The end item is only used for checkout (VICOS) resp. by the onboard software.

6.6.5.2.7.2 RESPONSE_PACKET

Defines a CCSDS packet issued in response to a SWOP Command.

Copyright per DIN 34

6.6.5.2.7.3 APPLICATION_ID (APID)

In COF/ISSA, the logical path between two addresses is defined by one application ID (APID). A pair of socalled CCSDS Endpoints is allocated to an APID. When sending data from one end point to another endpoint, the APID table defines the Application ID to be used.

6.6.5.2.7.4 CCSDS_END_POINT

In COF/ISSA, a logical path between two addresses on the CCSDS network is defined. Each address is called a CCSDS Endpoint. In the DB, each Endpoint is defined as an enditem, referencing the SWRU which is implementing the end point within the onboard system.

6.6.5.2.8 Acquisition Data Units

ADUs are the generic data packets received by CGS from SAS. They can contain different type of data:

- list of individual raw values
- a binary data block
- telemetry packets (CCSDS packets).

CGS/VICOS unpacks ADUs received from SAS and extracts raw values of measurements from them.

Depending on the type of data to be sent by SAS to CGS, different MDB enditem types describe the layout and data contents of the ADUs. These are:

a. Structured ADU description. If the ADU sent by SAS contains a list of raw values of measurements, then a structured ADU description has to used in MDB to describe the layout of the ADU.
In this case, the SAS will have to write individual raw values into the ADU. The ADU description tells the SAS and CGS how and where to put/get the raw values. This type of an ADU is helpful in case the SAS receives individual raw values from dedicated front–end equipment (such as voltmeters, data acquisition units or switching matrices) or computes certain raw values itself.

b. Unstructured ADU description. If the ADU sent by SAS contains a binary data block (except CCSDS packets, which are a special case, see below), then an unstructured ADU description has to be used in MDB to describe the layout of the ADU.
In this case, the SAS will have to write a binary data block into the ADU(e.g. a data buffer read from a front end equipment. The ADU description tells CGS how and where to put/get the raw values. This type of an ADU is useful in case the SAS receives blocks of data from dedicated front–end equipment (such as parallel IO boards, Mil–Bus interfaces or DMA type equipment) and the SAS shall not interpret the data, but has to simply forward it to CGS.

c. The CCSDS packet ADU description. If the data block received by a SAS is a CCSDS telemetry packet, i.e. a data block with a given internal structure, then this type of ADU applicable. The SAS has to put the right CCSDS packet (depending on CCSDS application id, packet type and packet id) into the ADU and CGS will extract the raw values of the measurements from this packet then.

The information in MDB describing an ADU (i.e. the ADU Description) is passed to SAS when CGS needs the data contained in it or if the user explicitly requests acquisition of these data.

The overall properties of ADU Descriptions comprise:

- a *general description* which is mandatory**.** This description contains some identification information which is needed for CGS and the SAS to properly handle the data packet

Copyright per DIN 34

- a *physical address*, which is optional. This property contains information needed by the SAS to physically acquire the data packet defined in the ADU and is not processed by CGS.
- a *measurement list* or *data buffer layout description* which is mandatory. This description tells CGS which measurements to extract from the ADU. In case the ADU is unstructured or CCSDS TM, additional information is provided in this aggregate to control where and how raw data have to be unpacked and formatted.
- a *description of the CCSDS (primary) header* which is optional. It is only needed for CCSDS packet ADUs and tells the SAS which CCSDS packet has to be put into the ADU.
- a *description of the CCSDS secondary header* which is optional. It is only needed of CCSDS packet ADUs and tells the SAS which CCSDS packet has to be put into the ADU.

6.6.5.2.8.1 STRUCTURED_ADU_DESCRIPTION

This end item describes the layout of a structured ADU.

6.6.5.2.8.2 UNSTRUCTURED_ADU_DESCRIPTION

This end item describes the layout of an unstructured ADU.

The definition of the start position of the different parameters can be ABSOLUTE or RELATIVE.

In ABSOLUTE mode of ADU the parameter start position is from the beginning of the packet starting with 1 (Bit 1 is the first Bit in packet).

In RELATIVE mode of the ADU the parameter position is derived from the end of the previos parameter or from the beginning of the packet. Here an offset is given for the space in bits between two parameters, offset 0 means immediately after the previous parameter.

6.6.5.2.8.3 CCSDS_ADU_DESCRIPTION

This end item describes the layout of a CCSDS TM packet ADU.

The definition of the start position of the different parameters can be ABSOLUTE or RELATIVE.

In ABSOLUTE mode of ADU the parameter start position is from the beginning of the packet part (Primary Header, Secondary Header, Data Part or Cecksum) starting with 1 (Bit 1 is the first Bit in packet part).

In RELATIVE modeof the ADU the parameter position is derived from the end of the previos parameter or from the beginning of the packet part. Here an offset is given for the space in bits between two parameters, offset 0 means immediately after the previous parameter.

6.6.5.2.8.4 PUS_ADU_DESCRIPTION

This end item describes the layout of an ADU containing a Packet Utilisation Standard (PUS) TM packet.

The definition of the start position of the different parameters can be ABSOLUTE or RELATIVE.

In ABSOLUTE mode of ADU the parameter start position is from the beginning of the packet part (Primary Header, Secondary Header, Data Part or Cecksum) starting with 1 (Bit 1 is the first Bit in packet part).

In RELATIVE modeof the ADU the parameter position is derived from the end of the previos parameter or from the beginning of the packet part. Here an offset is given for the space in bits between two parameters, offset 0 means immediately after the previous parameter.

Copyright per DIN 34

6.6.5.2.9 Simulated Data

In order to support the simulation mode on a test node, it is possible to define simulated data in the MDB for ADUs and measurements in a dedicated enditem type. These predefined data will be used as initial values when running in simulation mode.

6.6.5.2.9.1 SIMULATED_ADU_DESCRIPTION

This is an end item describing the contents of an ADU (and their related measurements) in terms of simulated data.

6.6.5.2.10 Lists

It is often useful to group certain types of end items in lists. For this purpose, MDB provides two predefined lists, being:

- Monitor lists (Measurement/SW Variable/Derived Value Lists)
- PUS Structure ID (Measurements with PUS Structure Identifier)
- Stimuli lists (GDU Description Lists)

Monitor lists can be used to collectively control groups of measurements, e.g. to enable them for monitoring, start_acquisition or enable processing Thus they can only contain references to other MDB items of type measurement, SW variable or Derived Values.

Stimuli lists can be used to define a predefined sequence of stimuli if this sequence is often used operationally. Thus stimuli lists can only contain references to other MDB items of type stimuli. The list has two parameters, FIRST and LAST, which at runtime or by MDB default values control which subsection of the list shall be used (e.g. a stimulus list contains references 1..100 to other stimuli and it is invoked with actual parameters FIRST=10 and LAST=20, then only elements 10..20 will be processed).

The overall aggregate structure of the list enditems is similar. They contain one aggregate which defines the list and only the stimuli lists contain another aggregate which defines the formal parameter list.

6.6.5.2.10.1 EGSE_MONITOR_LIST

This end item defines a list of measurements and/or SW variables/Derived Values.

The structure of this enditem is rather simple and it contains only one aggregate.

6.6.5.2.10.2 PUS STRUCTURE_ID

This end item describes the layout of a structure as defined in the Packet Utilisation Standard (PUS).

A structure ID is downlinked in the TM packet and identifies the parameters associated with it.

In CGS, a Structure ID may be referenced in an PUS_ADU_DESCRIPTION and thus define the layout of an ADU containing a PUS TM Packet.

6.6.5.2.10.3 GDU_DESCRIPTION_LIST

This end item defines a list of stimuli or telecommands. Such a list can be executed in the online system giving a single ISSUE statement.

Copyright per DIN 34

6.7 Consistency Checking

6.7.1 Input Checking

Some items (e.g. value ranges) are checked already when entering the data in the DB input masks. In case the data entered is not valid, a beep together with an error message is generated and the data is not accepted.

6.7.2 Item Checking

6.7.2.1 Consistency Checker on End Item Level

The consistency checker (CC) may be called on end item level to verify consistency rules on each enditem in the currently selected scope (CDU,CCU). The CC is called via the Flexible Tool Interface (Under “Tools”) Refer to MDA Reference Manual [2.1.1.3] for description of the CC.

For alist of checks performed and their classification see Appendix N.

6.7.2.2 Check_MDB_Item Program

For some items. a tool Check_MDB_Item may be called via the flexible tool interface (”Tools” Menu for the enditem).

The Check_MDB_Item program is implemented for the following enditem types:

MEAS	EGSE_xxx_MEASUREMENT
	EGSE_xxx_SW_VARIABLE
	EGSE_xxx_DERIVED_VALUE
ADU	xxx_ADU_DESCRIPTION
GDU	EGSE_PREDEFINED_TC
	EGSE_BINARY_PACKET
	EGSE_ANALOG_STIMULUS
	EGSE_DISCRETE_STIMULUS
	PUS_TC
SYNOPT	WDU_GROUND_SYNOPTIC_DISPLAY

Copyright per DIN 34

The program performs predefined checks and displays the result to the user by opening a text window.
In this window the errors found, a statistic on errors found and the enditem data loaded from the MDB is
displayed.

In case there are errors reported by the loading software, a specific error log file is opened showing the
error messages.

The program also searches for references in other enditems of the scope to the item under check. These are currently
– references of measurements/sw_variables/derived values to other measurements/sw_variables/
derived values via conditions
– references of synoptics to measurements/sw_variables/derived values
– references of ADU_DESCRIPTIONs to measurements

Note: The searching for references may take some time in large databases. Therefore the searching is to be suppressed via the environment variable MDA_CHECK_MDB_ITEM wich is set in scipt `$MDA_HOME/bin/sun5/check_mdb_item`

For a list of checks performed and their classification see Appendix N.

6.7.3 Consistency Checking on CDU/CCU Level

The consistency of a CDU or a CCU is checked via the ”Consistency Checker (CC)” Program. It may be called via the ”Command” menu of a CDU resp. the ”Command” menu of a CCU Version.

The CC defines the consistency status of a CDU/CCU.

For description of the CC see MDA Reference Manual [2.1.1.3]

6.7.4 Checking when Loading to Files

When creating the SCOE load files, the MDB data are checked within the scope of the loaded data again.
Errors are reported to the user. (Refer to ch. 7.1.11). Also each test node verifies that data loaded is consistent with the constraints for the testnode.

Copyright per DIN 34

6.8 CIS Interface

The CGS Interface Server (CIS) provides a new Corba interface for sending telecommands, events logging, HLCL commanding and measurement requests to CGS. The interface includes an Corba IDL usable at the external user's side. The CIS implements the Corba Server. The CIS interface is defined in section 2.1.3.

The CIS provides the following major services:

- **Identification of Connected Application / Application Session Creation**

  The client application needs to identify itself with its user's name

- **Logging of Events**

  The client application can log an event message to CGS. This event will stored always into the T(est) R(esult) D(ata)B(ase) and, depend on a parameter, send to the message services / message handler.

- **Access to CGS Events**

  The client application may connect to an event service, which distributes selected events to application when they arrive at the CIS. For a report of events a XML schema is defined (see 6.8.1).

- **HLCL Commanding**

  Any HLCL command may be executed via the CIS. The execution will take into account the user's privileges according to the identified role.

- **Access to Measurements / SW Variables / Derived Values**

  A list of measurements / SW Variables and Derived Values can be requested from the CIS. The CIS will send the engineering values together with the acquisition status to the requesting application.

- **Provide Entry to the Command History**

  A new or updated entry (telecommand and/or additional information) to the command history is sent from the client application to the CIS, which stores it into the Command History of CGS.

- **Get Command History Information**

  The client application can access the Command History information, stored within the Command History. A list of commands together with additional information is provided according to a given filter.

  The client application may subscribe to get updates of a specific Telecommand or all Telecommands sent from a specific node after a specific time.

  For a report of Command History information a XML schema is defined (see 6.8.1).

- **Prepare a Telecommand Packet**

  The CIS can prepare a binary Telecommand Packet for different packet types:
  SWOP commands
  FLAP commands
  Arm and Fire commands
  Ordinary Telecommands
  This packets will be returned to client application without sending.

- **Send a Prepared Telecommand Packet**

  A prepared binary Telecommand packet can be sent through the CIS to the addressed S(pecial) A(pplication) S(oftware).

- **Get Reference to Used CCU in MDB**

  A reference to the CCU internal identifier and the consistency date of this CCU (i.e. the date of the latest change in CCU scope) is provided at login time. Each configuration change causes a notification to the client.

Copyright per DIN 34

The CIS is to be defined as one of the CGS Processes, with identification in the System Topology Table. It needs to define in the System Topology Table and in the M(ission) D(ata) B(ase) too. If the CIS is part of a test configuration, it will be started automaticly during test configuration startup.

The formal CIS interface specification is referenced in appendix 0.

6.8.1 CIS XML Schema Definitions

The CIS schema definitions are covered by the following files:

- logEvent.xsd
- CommandHistory.xsd

The XML schemata are available at CGS directory location $CGS_HOME/gsaf/lib/xml/cis_reports.

Copyright per DIN 34

7 INTEGRATION AND TEST

7.1 General Operation in the Checkout Environment

7.1.1 Checkout Operations

During checkout operations, CGS allows to monitor and command the Unit under Test and to simulate the environmet of the UUT or the EGSE. The DB servers, testnodes and workstations are active, the data is acquired via SAS and Frontends and the Simulation Model might be activated to simulate data via frontends to the UUT. Commands are given to the UUT/Onboard System and to the EGSE, reconfiguration requests are performed and acquired data stored in the TRDB during the test can be evaluated in parallel to ongoing test operations or offline after the test.

7.1.2 Operational Modes

CGS allows to operate for checkout in different modes:

- **EGSE NORMAL** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode NORMAL
  SAS are connected to front ends. Front ends are active.
  SAS and Automated Procedures may run to access online data. (*Not used in current version*)
  Simulation on Simulation Nodes might be active or not. (*Not used in current version*)
  Simulation can be controlled via a specific window (MOCS) or via the HCI/HLCL Command Window (”Central Commanding”) (*Not used in current version*)
  TEV may be used to evaluate data online
- **EGSE Simulation** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode SIMULATION. Data from frontends is internally simulated by test nodes. Commands are not sent.
  No SAS connecting to front ends are running . Front ends are inactive.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes is not active. (Not used in current version)
  TEV may be used to evaluate data online
- **EGSE ”Mixed”** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode NORMAL or SIMULATION.
  On test nodes in NORMAL mode: SAS are connected to front ends. Front ends are active.
  On test nodes in SIMULATION mode: No SAS connecting to front ends are running . Front ends are inactive.Data from frontends is internally simulated by test nodes. Commands are not sent.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes might be active or not. (*Not used in current version*)
  Simulation can be controlled via a specific window (MOCS) or via the HCI/HLCL Command Window (”Central Commanding”) (*Not used in current version*)
  TEV may be used to evaluate data online

Copyright per DIN 34

Another EGSE ”Mixed” Mode is the operation of test nodes in REPLAY in parallel to test nodes in SIMULATION.

- **EGSE Replay** mode
  EGSE is online and acquiring data from the UUT. Commands are sent to the UUT
  Test Nodes are operating in mode REPLAY
  No SAS are connected to front ends. Front ends are inactive.
  SAS and Automated Procedures may run to access online data.
  Simulation on Simulation Nodes is not active. (Not used in current version)
- **Standalone Simulation** mode (*Not used in current version*)
  Simulation on Simulation Nodes is active.
  Simulation is controlled via a specific window (MOCS)
  EGSE is not online .
  Test Nodes are not operating. HCI on workstations are not operating.
  No SAS are connected to front ends. Front ends are inactive.
  No SAS and Automated Procedures are running to access online data.
  TEV may be used to evaluate data offline after the simulation run.
- **Offline Evaluation** mode
  TEV is used to evaluate data offline.
  Simulation on Simulation Nodes is not active.
  EGSE is not online .
  Test Nodes are not operating.
  No SAS are connected to front ends. Front ends are inactive.
  No SAS and Automated Procedures are running to access online data.

7.1.3 Operational Configurations

For the different modes, CGS can be configured in different operational configurations.
An operational configuration is determined by

– The **CGS System Services** runnning
The CGS System Services consist of
  – The Time Services
  – The Services to startup processes on local or remote nodes
  – The TRDB Processes (DBS)

– The **CGS Applications** running
The CGS Applications consist of:
  – The processes HCI, TES,TEV,TSCV, CIS (group of processes)
  – other interactive tools (I_MDB, DDED, CLS Editor, GWDU etc.)

Copyright per DIN 34

– The **Test Configuration** loaded
A Test Configuration determines the workstations, test nodes and simulation nodes involved, the mode of the nodes and the data to be loaded.
A test configuration is defined in the MDB, loaded and executed by the TSCV tool.
As it is part of a specific CCU of the MDB, activation of a test configuration includes selection of a CCU for a test.

The minimum setup for the operational modes are as follows:

For all: Oracle Services and the CGS daemon process need to be running. They are started at boot time automatically.

Dependent on the operational modes, additional system services and applications are requried. They are setup using the TSCV program or (for system services) the task selector's menu.

- **EGSE NORMAL, Simulation** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **EGSE Simulation** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstations in parallel
- **EGSE Replay** mode
  – All System Services are running
  – Applications on Test Nodes (TES) and Workstations (HCI) need to be runing
  – The Message Window is required
  – TSCV is required to setup the configuration
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **Standalone Simulation** mode
  – All System Services are running
  – Applications on Simulation Nodes (CSS Group) need to be runing
  – Workstations: the MOCS windows allow to control the simulation, no HCI is necessary.
  – The Message Window is required
  – Interactive Tools (incl. TEV) may be used at any workstation in parallel
- **Offline Evaluation** mode
  – Only the DBS Processes must be setup (Can be done via task selector)
  – TEV is running on workstation(s)
  – The Message Window is required
  – Other Interactive Tools may be used at any workstations in parallel

Copyright per DIN 34

7.1.4 Operational Constraints

In the following, a list of constraints is given, to describe what must be considered in the setup or during the checkout operation.

Test Configurations

- A Test Configuration (TC) is loaded by TSCV into one of 5 predefined slots.
- Each TC has a CCU associated. By definition, this is the CCU where the TC is loaded from.
- TC define the role of each node for a test: It defines, which nodes are participating, and which test node is the Master Test Processor (MTP).
- Each Test Configuration has a test session associated, where all results generated within the test configuration are collected. If no specific name is given, the ”Default Test Session” is selected.
- Up to 5 Test Configurations may be setup in parallel, each having different test nodes and workstations defined, but all having the same DB Server node and operating on the same TRDB. The result must go to different (named) test sessions, except one, which can go to the ”Default Test Session”.
- Test Sessions can be specified before the Test Configuration is set to active. Then the test session is created/opened during the setup.
  Test Sessions may be created and closed, however, at any time when a test configuration is active.
  A test session is always allocated to one specific test configuration, except for the Default Test Session, which is open always, also when no test configuration is active.
- Each test session may be setup with Final Archiving active or not. Final Archiving active ensures, that data is written to the Optical Disk during test operations, whenever a predefined threshold for disk usage is reached. The threashold is configurable in the DBS Configuration files.

  Two Optical Disk drives are supported for final archiving. I.e. a maximum of two test sessions may have Final Archiving active in parallel.

  When Final Archiving is enabled, the device media should be ready and mounted on the drives before the test session is created.
- A test configuration may be modified online in TSCV.
- Test Evaluation (via TEV) and Standalone Simulation can be performed without having a Test Configuration setup.

EGSE Normal Mode

- Before setting up the Test Configuration, it has to be ensured, that all frontends are booted up and in their correct state to be able to connect to CGS test nodes.

EGSE ”Mixed” Mode

- A mixed configuration of test nodes in different modes might be used to verify a part of the UUT/EGSE and simulating another part. There is no constraint on the number of nodes being in either mode.

Copyright per DIN 34

EGSE Replay Mode

- Test Sessions replayed must be selected via TSCV. They must exist in the TRDB and the data must be online (i.e. not exported/archived to optical disk).
  The used Test Configuration must exist in the MDB.
- Before a test session is started for replay, it must be ensured, that the test node's local disks have enough capacity to take all archive files needed for the selected time frame.
  It may take a considerable amount of time to distribute big archive files to the local disk of the test nodes, before the test session can be started for replay.
- In replay mode, the synchronisation of all test nodes with respect to the data replayed is not fully guaranteed. Test Nodes are setup in the sequence they appear in the Test Configuration. The start commands are given to the test nodes with a delay of roughly 100 ms between them.
- In replay mode, when replaying data, it can happen, that messages generated from replay data are received by the message window in a slightly different order, especially when a high load is selected by increasing the replay speed factor (see below)

Copyright per DIN 34

7.2 Setting-Up the Test Environment

7.2.1 Introduction

This section consists of 3 parts. The first part provides a detailed user overview of the Test System Configuration and Verification (TSCV) software which is the primary tool used in setting–up the test environment. The second part shows by example how one can configure the generic test system to user specific test environment and start a test execution session using TSCV. The third part explains how to setup automatic archiving of an execution session to the Final Archive medium.

7.2.2 Test System Configuration and Verification (TSCV)

The role of TSCV within CGS is to prepare the EGSE for a test session, implying

– select and load Test Configuration templates (up to five) from the database
– initiate set–up with one of the loaded test configurations as template
– ensure that the DBS processes are up and running
– Set the global default CCU in MDB for an active test configuration
– ensure that the TES processes on the testnodes are up and running
– command the testnodes' test execution software (TES) to initialize and load configuration data from the Mission Database according to the Test Configuration template
– Notify HCI
– control the time services system (TSS) initialization
– command the testnodes with respect to their overall state (execution mode)
– initiate (open) and close test session in the Test Result DB
– distribute archived test result files to the testnodes for the purpose of test replays
– initialize and maintain the Test Result DB
– monitor the status of the test nodes
– maintain system configuration files for the system topology

The TSCV product runs on a CGS workstation, and can execute in two modes: either as a batch program which executes its task solely on basis of data provided as parameters on the command line, or an interactive program.

7.2.2.1 Selected CCU Version

The access ways to data in the configuration database is like a directed graph (with nodes of varying types). The database concept is such that all configuration data prepared for a test must be collected under one and the same access node, which shall be of type CCU. TSCV lets the user select such a node, and uses this selected CCU as a 'root' address in the access tree.

The notion of version is also used in the configuration database, i.e. any node in the access graph may appear in several versions. The user therefore not only must select a CCU node; a specific version of that node must also be selected.

Copyright per DIN 34

7.2.2.2 Describing and Configuring a Test Configuration

A “Test Configuration” is defined as a end item type instance (EGSE_TEST_CONFIGURATION) in the MDB. A test configuration describes resources, services and data involved in a specific test and control task that need to be distributed to a set of test nodes.

The view of a certain test configuration in the MDB and the scope of data that can be referenced by a specific test configuration is defined by selecting a CCU in the MDB. A test configuration may use different CDUs within the scope of a CCU.

A test configuration that is defined as an end item type instance in the MDB can be exported a SCOE file that has to be loaded by each test node contributing to a test.

The mapping of resources and services for a specific test session (physical workstation, logical host name) is defined by a system topology table as described later.

The SCOE files and the system topology table are part of the setup data to describe a test configuration that can be allocated to a test session. Each participating test node requires a dedicated SCOE file.

The TSCV main interface window allows up to five simultaneously loaded test configurations shown in a dedicated test configuration panel. A loaded test configuration is allocated to a dedicated test configuration slot in the second panel in the TSCV main wiondow. Selecting a test configuration from this test configuration slot defines the view of the third and forth panel with further information of a selected test configuration.

Simultaneous test configurations correspond to concept, that the nodes in CGS network can *cooperate in groups*. Each node on the CGS network may or may not be in one group of nodes which participates to execute a certain test activity. At any time, the CGS network may be divided into 5 cooperating groups (plus the group which consists of the nodes which are not part of any group). The term Test Configuration is used also for these groups. The loaded test configurations can perform testing/test execution concurrently.

For each test configurations mode changing controls and different control actions can be performed.

TSCV can also set–up the test system to replay a recorded test session, i.e. data stored in the test result database during a former test session. The test configuration used for replay set–up may be the same as applied during the recording. However, it is also possible to associate an existing session with a test configuration already loaded from the mission database. The replayed data sets are then mapped to replaying test nodes based on the MDB pathname of the recorder and the replayer.

7.2.2.3 Setup a Test Configuration

It may or may not be a direct correspondence between a specific group on the CGS network and a Test Configuration that TSCV manages. If a Test Configuration do correspond to a group, it is said to be ’executing’. To bring a loaded Test Configuration into use (i.e. to the state active), the user must command TSCV to execute set–up.

However, before executing Set–up, the user has the option to ’edit’ a loaded Test Configuration. It is possible to specify that even though a node appears in the Test Configuration template, it shall not participate in the corresponding group. Further, there are some attributes and parameters associated with the nodes that TSCV uses during the Set–up function that are set when loading a Test Configuration, but which the user can change.

After editing the Test Configuration, the user can give a command to ’Execute Setup’. This implies, among other things, to send a command to the testnodes requesting initialization (the selected test execution mode is sent with this command). It also implies to control initialization of TSS.

Copyright per DIN 34

A test configuration can be active or idle.

Figure 7–1 below shows an example of an EGSE set–up with three *Active* Test Configurations (groups).

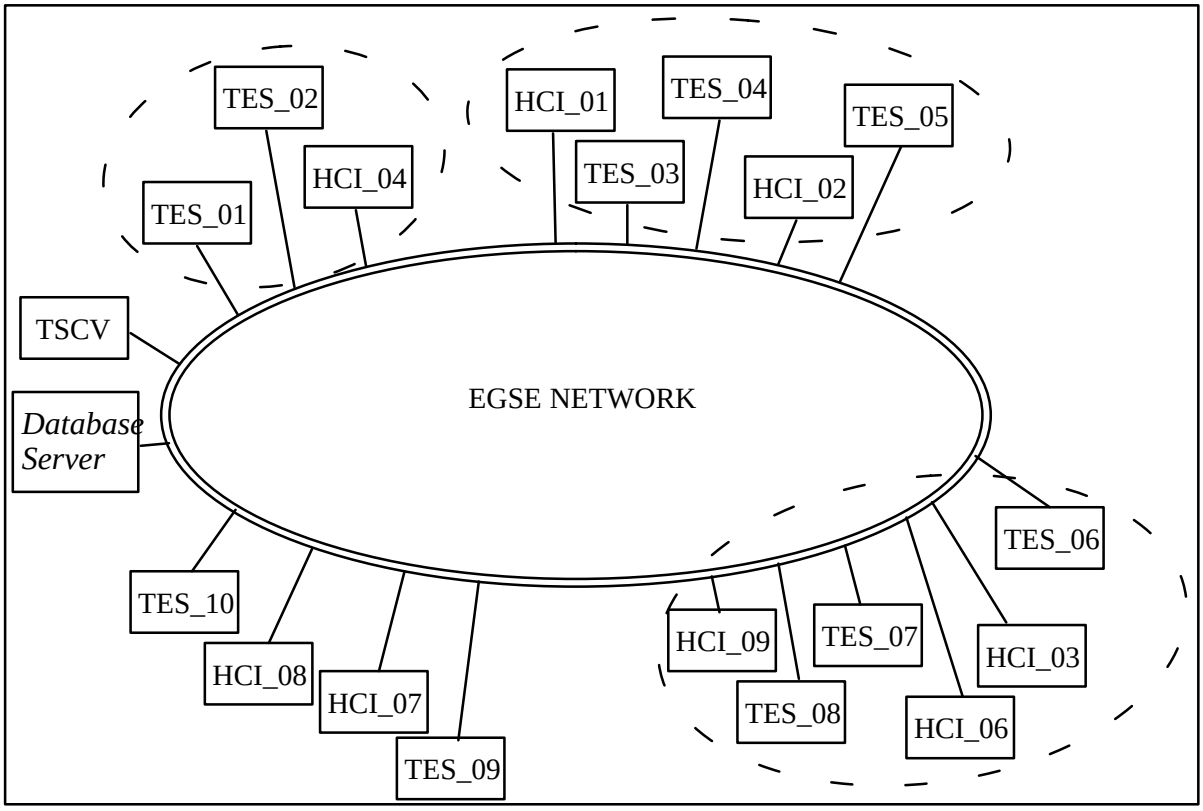


Figure 7–1 : *Example of a Set–Up with three Active Test Configurations*

7.2.2.4 Test Node States and Control

The user may use TSCV to send commands to the test nodes to start, suspend, resume, and to stop testing. These functions will act upon one test node or a set of test nodes that the user has selected from one of the Test Configurations. The state transitions of a test node are shown in the Figure 7–2 below.

Copyright per DIN 34

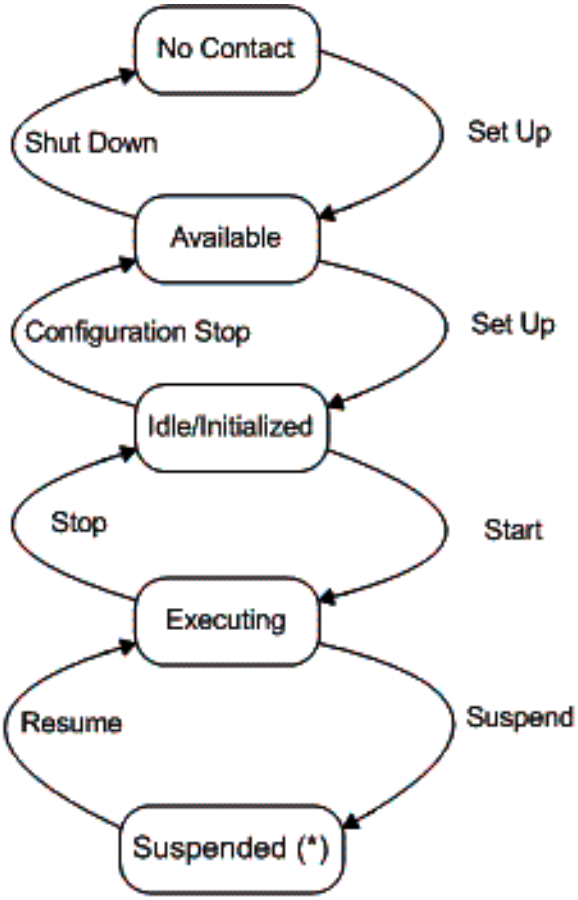


Figure 7–2 *: Test Node State Transitions*

Remark: (*) The test node state “Suspended” is only available in simulation mode. State transitions from “Idle”, “Executing” and “Suspended” states to “No Contact” state are also posible by a shut down operation.

7.2.2.5 Test Session States and Control

A Test Execution Session may be regarded as an envelope into which all test data generated during a test are inserted. Before starting testing in one of the groups within the CGS network, TSCV cooperates with the user to create/open a uniquely named test session.

A single test configurations can be allocated to a single test sessions. Test configurations that are not allocated to test sessions are allocated to a default test session.

A non–default test session can be in the state active/open, closed, aborted, exported or archived as shown in Figure 7–3. Test session state transitions are supported by different CGS tools as indicated on the figure.

Copyright per DIN 34

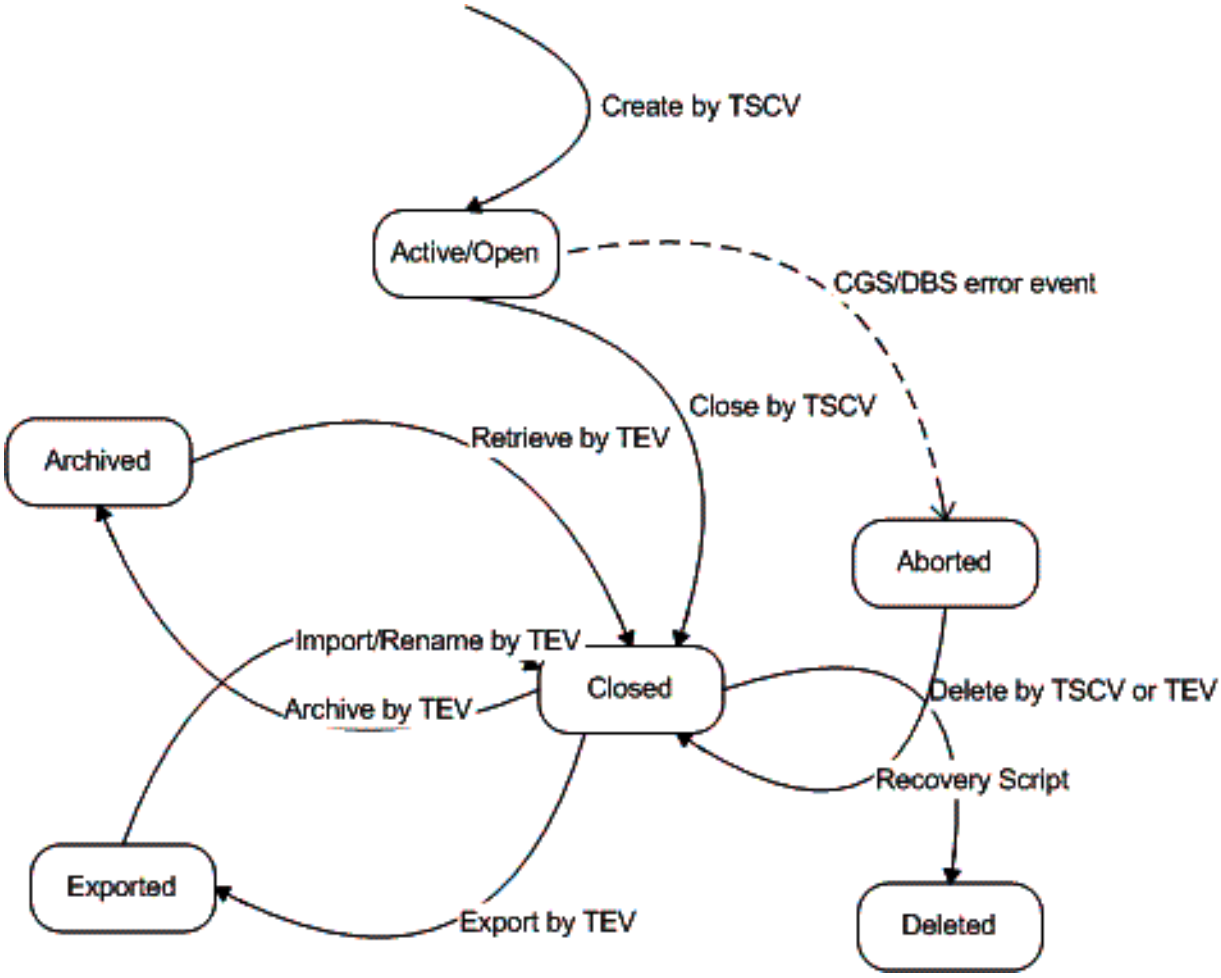


Figure 7–3 *: Test Session State Transitions*

Test sessions may be scheduled in a *cyclic* way by a specific CGS application. A cyclic test session has to be specified by a cycle time and a start time.

Test sessions are logged in a Test Result Database (TRDB). Logging results are stored in the TRDB as ORACLE tables and the CGS files.

- *Critical memory resources are identified in chapter 10.*

There will be one such session envelope for each Test Configuration which is executing, i.e. for each group on the network. Hence, at any time there may be up to five simultaneously open test sessions in the test result database.

The Test Result Database manager will accept test–data also from a group of test nodes for which there is no open test session allocated. These data data are allocated to a single Default Test Session.

7.2.2.5.1 Deleting Test Sessions

The user can select a test session from the Test Result Database and to command the Test Result Database manager to delete that session. The envelope as well as its content will then be deleted. The envelope is represented by the key record created at session opening.

Copyright per DIN 34

7.2.2.6 System Table Maintenance

TSCV also provides access to a few essential system tables. A control interface is provided to maintain the SYSTEM_TOPOLOGY_TABLE, while a view is provided of the VERSION_ID_TABLE (installed software versions).

7.2.2.7 System Services Management

The test conductor can control a few essential system services with TSCV. It will be ensured that the central DBS applications are up and running at TSCV startup, during setup of a test configuration and when checking status for a test configuration.

Further, the Time System Services which provide the Simulated Mission Time will be started and set–up as necessary.

Also the CGS basic services running on the different CGS computers may be started or shutdown with TSCV.

7.2.2.8 Operations Environment

TSCV's Configuration

An example of the TSCV configuration is found in appendix K.

7.2.2.8.1 TSCV Operations Constraints

The TSCV configuration file is stored in *$CGS_HOME/local/config*. In particular, this directory will contain data which reflect the system state as it was when TSCV was last terminated.

The user who shall run TSCV must have the system privilege "CGS:SETUP" (use the priv–tool to evaluate).

See section 10 for description of user roles and privileges.

TSCV cannot be executed unless the OpenWindow system is running. This applies also for the batch operating mode when automatic final archiving is used. .

7.2.2.9 Operation Basics

TSCV provides a set of functions related to administering Test Configurations and set–up CGS according to configuration templates, to monitor system state, to start and stop testing, to open and close test sessions.

TSCV has two command modes:

- The interactive mode where the user commands TSCV interactively by manipulating Graphical User Interface (GUI) items such as buttons, textfields, lists and menus.
- BATCH mode, where TSCV takes as input data provided as command line parameters and executes set–up, starts testing, and terminates when the test is complete.

For reference information on how to use TSCV version 4.1, refer to Chapter 9. Basic interaction techniques are not described, such descriptions may be found in the Open Look documentation.

In the following, an overview presentation is provided of what can be done with and how to operate TSCV. The presentation follows the structure of the main TSCV window. This is horizontally divided in three sections. The upper part relates to the test system as such, i.e. corresponding to basic CGS services. The middle part is where the user mainly will operate. It consists of a list representing the test configurations managed

Copyright per DIN 34

with TSCV, together with menu buttons representing functions which my be applied on these. The lower part of the window presents details of that test configuration which at any time is selected in the test configuration list, i.e. details relating to each of the applications associated with the test configuration. The lower part has two menu buttons, providing the possibility to control each node separately.

7.2.2.9.1 Test System Control and Configuration

Test system maintenance (test configuration and test session maintenance) tasks are dedicated to the upper panels of the TSCV main window.

The test system, i.e. the various processes making up the test system, may be started and stopped, and a menu is provided to activate these functions.

When activating the Start function, a script which is part of the CGS will be activated. This script will start–up all the CGS services on all the test system computers. The Shutdown function (**?**) will accordingly invoke a shutdown–script, but before that script is invoked, any test activity will be stopped and resources allocated to any active test will be released.

Certain properties of the test system may be viewed and edited. This is properties represented by the SYSTEM_TOPOLOGY_TABLE and the VERSION_ID_TABLE. The latter may only be viewed, while the others may be edited.

The SYSTEM_TOPOLOGY_TABLE is a mapping between physical application instances and hosts. Functionality is provided to add, delete and modify application entries. Before making the user's changes applicable, TSCV will check a range of consistency constraints. Each CGS component that participates in a test session are configured by this table during start–up.

- *A running test system (test configuration and test session) should be stopped and restarted after changes in the system topology.*

TSCV automatically provides the user with the option to shutdown the whole system, including TSCV itself.

7.2.2.9.2 Test Configuration Management

The main task for TSCV is to provide control of test configuration set–up. This concept is described in section 4. Up to five test system set–ups (groups of cooperating test resources) may be managed in parallel. Each such set–up is represented by a 'slot' within TSCV main window. Each slot may be filled with a test configuration table, which serves as a template for set–up. When test system resources are allocated to a slot, it is said to be **Active**. An **Idle** test configuration slot is a template for a potential set–up.

The test configuration slots are represented by a list in the central part of TSCV's main window, which thus has a fixed set of five entries, mapping slots maintained by TSCV. In the list, some key data is provided for each slot, according to the list header: test configuration name, whether the status is **Idle** or **Active**, and the name of a test execution session which may be associated to the slot. In addition, there is space for the name of a session to be replayed – which will be applicable only for replay set–up. When starting TSCV, the same set of test configurations will appear as when last stopped. After installing TSCV, the list will of course be empty. By functions available through the **Edit** menu, the slots may be filled or emptied. Filling a slot implies to load a test configuration (thus, to empty is to **unload**). Normally, load will be from the mission database – corresponding to an option in the **Edit** menu. For replay purpose, it is also possible to fill a slot with a test configuration loaded from the test result database.

All TSCV functions, except for the functions described under sec. 7.2.2.9.1, addresses the test configuration slot which at any time is selected in the list (it is only possible to select one entry at a time).

Copyright per DIN 34

7.2.2.9.3 Test Configuration Control

Essentially, TSCV's purpose is to set–up test configurations. This means to allocate and prepare resources in the test system to be used for a test, where a test configuration table (together with the SYSTEM_TOPOLOGY_TABLE) is used as a template.

This set–up control is available via the menu button **Configuration**. To set–up a test configuration implies to check that the resources prescribed by the test configuration table are available and not allocated to another test, to allocate the resources and to control the set–up of each individual resource. The following happens when the **Setup** option of the Configuration menu is activated:

- A session is allocated in the test result database. The user may in advance have used the function for creating a session in the slot, in which case TSCV will open that session in the test result database. Otherwise, the database's default session will be allocated. (If already allocated by another test, the set–up will be interrupted). First it is checked whether the central database processes are up and running, and if not, the DBS start–up script will automatically be started.
- The mission database is set–up such that the CCU identifier embedded in the loaded test configuration becomes 'global default' for the test identified by TSCV's slot number.
- The HCIs of the test configuration are allocated, i.e. a message is sent to each application, signalling that it now has become part of an active test configuration. (If already allocated by another test, the set–up will be interrupted).
- The Time Synchronization Services is set–up, i.e. an SMT domain is set–up for the test (the domain will comprise the TSS processes on all hosts affected by the test configuration set–up). First it is checked whether the TSS processes are up and running, and if not the TSS start–up script will automatically be started.
- The TES'es of the test configuration are allocated, i.e. each application instance is commanded to perform initialization. (If already allocated by another test, the set–up will be interrupted). First it is checked whether the TES processes are up and running, and if not, the TES start–up script will automatically be started.
- Optionally, if a replay is prepared, recorded data will be retrieved from the test result database and distributed to the test nodes.

After completion of the set–up, the test configuration and the corresponding TSCV slot is considered **Active**.

To **Start** a test configuration, a corresponding option in the **Configuration** menu should be activated. TSCV will then interact with the allocated test nodes and command them to change from the state **Idle** to **Running**. From that time, they are left to be controlled by HCI. If Start is activated on a test configuration/slot in the state **Idle**, the test configuration set–up will be performed implicitly before sending initializing commands to the test nodes.

To deactivate an active test configuration/slot, the menu option **Stop** should be applied. All test nodes will then be commanded to stop test execution. Furthermore, the test session opened in the test result database will be closed, and the Time Synchronization System will be commanded to clear the allocated SMT domain. All test system resources allocated to the test are released.

A test configuration slot may also be **Shutdown**. By activating this function via the menu, all applications appearing in the slot's test configuration table will be shut–down. If applied on an active test configuration slot, each executing test node will be stopped implicitly before shutting down each participating application. The test session will not be closed, and the test configuration will not changes state. The user will be requested

Copyright per DIN 34

to choose whether or not to include the basic CGS services on the hosts affected by the shutdown (node shutdown).

An active test configuration where one or more of the test nodes are in simulation or replay mode may be commanded to **Suspend** and **Resume** testing. By activating this function with the corresponding menu options, TSCV will interact with those simulating nodes and command them to change their internal state correspondingly.

TSCV also provides the option to **Check** the status of a test configuration. When the corresponding menu option is activated, TSCV will interrogate the active resources used by the test configuration. If the test configuration state is **Idle**, DBS the TES instances are addressed. Otherwise, also each TSS process allocated to the test configuration will be interrogated. If DBS or TSS does not respond, TSCV will offer the user to start the processes. For the TES instances, only the displayed status will be updated, and should a TES instance be unavailable, the user may launch that individual application and set it up. This is provided via a menu in the lower part of the TSCV main window. By the way, TSCV will periodically and automatically (with a configurable frequency) check the status of the test nodes used by an active test configuration, provided it is in focus.

7.2.2.9.4 Test Configuration Edit

It is the content of the test configuration slots which may be edited – provided that no test system resources have been allocated to the selected slot (i.e. that the slot is **Idle**). There is a separate menu for this.

Normally, a test configuration to be used for set–up will be loaded into the slot from the mission database. It is also possible to set–up such that a formerly recorded test may be replayed. In that case the slot may be loaded with the test configuration entity embedded in the session – which is a copy of the one used during the recording session. There are two corresponding menu options provided. Loading can of course not be done if the selected slot is **Active**. Possible existing data will be overwritten when loading.

Before loading, a test configuration has to be selected – either from the mission database or from the test result database. Thus, the menu options for loading will open separate windows.

The **Load for online test** option is associated with a window labeled **Load Test Configuration for Online Test**, which will list test configuration entities available in the database – for selection and loading. The list contains the entities belonging to one CCU version. There is always a default CCU applicable when the window is opened (normally the same as when the window last was closed). If another shall apply, it must first be selected in another window – opened from the load window. The CCU selected for loading will also apply for the set–up (the test system resources will be informed about the applicable CCU during the set–up process).

The **Load for replay** option is associated with a window labelled **Load Replay Session to be Replayed**. As the window title suggests, loading from the test result database and selection of a session to be replayed is one operation. Thus, in the main window's slot list, the session to be replayed will be identified – and this is the immediate visual feedback that a slot is for replay (regardless the Active/Idle state). TSCV will ensure that the loaded test configuration entity still is existing in the database (otherwise it will not be possible to replay the session). Note, however, that it is not a requirement to replay a test that a frozen version of the test configuration was applied.

Provided the selected slot is not empty, its content – i.e. the test configuration table details – may be viewed. It is also possible to empty a slot – **Unload** a test configuration – provided it is Idle .

Copyright per DIN 34

7.2.2.9.5 Test Session Control

During a test, all resources allocated to a slot will enter data into the test result database, to be collected into an entity called test session. DBS always provides a default test session, into which the data will go if not a separate session has been allocated (created) for the test.

- *It is recommended to allocate session names to test sessions.*

The user may create and allocate a test execution session with the selected test configuration slot. A menu option which opens a window labelled **Create Session** is provided for this. When creating a session in the test result database, TSCV will provide DBS with information about which applications are participating in that test, such that it may direct received data to the right session. Accordingly, the **Close** option will deallocate the session from the slot (and DBS will prevent that any more data is entered into it).

TSCV imposes the constraint that the default test session at any time may be allocated to maximum one active test configuration slot. This constraint is checked before setting up a test configuration slot, and before closing a session. There is also a constraint that only active test configurations may allocate a session in the test result database. Thus, the **Create** and **Close** options have a different effect if applied on an Idle slot; they provide information to TSCV which will be used when the slot subsequently is set–up (to actually create the session in the database and allocate it to the slot).

The **Maintain** option provides access to the **Maintain Test Session** window, in which the user can apply various selection criteria to select subsets of sessions from the test result database. The selected sessions are provided in a list, and individual list entries may be deleted. A selected session may also be closed if it has the status open and is not displayed in the main window of TSCV.

7.2.2.9.6 Replay Session Control

Assign/Load Session to a slot

As described in sec. 7.2.2.9.4, a recorded session is implicitly associated with the slot as a replay session, when loading a test configuration from the test result database. It is, however, also possible to make such an association to a slot containing a test configuration loaded from the mission database. A menu option **Assign** is provided for this, and will open a window labelled **Assign Replay Session to Test Configuration** (which is almost identical to the **Load Replay Session to be Replayed** and **Maintain Test Session** windows). The assigned session will appear in the slot list under the column header Replay Session.

Role of Master Test Processor (MTP) during Replay

It is mandatory to have one testnode acting as the MTP during replay mode. The MTP is the only node reading the time control records from archiving files and setting up the SMT as it was during online session.

The MTP is responsible for maintenance of the time base in the CGS system. Therefore it is recommended for a replay session, to have archive data of the MTP. Otherwise it has to be accepted, that no valid SMT replay time is available.

Note: In case no data has been recorded by the MTP during the test, it is possible for the replay session to declare another test node as MTP during the replay session setup.

Select Time Frame

When a replay session is associated to a slot (regardless whether ’loaded’ or ’assigned’), the menu option **Properties** becomes active – in view or edit mode, depending on whether the slot is **Active** or **Idle**. That

Copyright per DIN 34

window will display the time when the recorded session was created and closed. By default, that time frame will be replayed, but in edit mode it is possible to restrict the time interval (by specifying begin and end time). The time base may be switched between Local Time and Simulated Mission Time, and the time frame may be specified by setting the date and time directly. Otherwise, the user events logged during recording is provided in a list, and the begin and end times may be set by selecting entries in that list. The specified time–frame will be sent to the test nodes when the slot is set–up, and TSCV will itself use it when copying data to be replayed from the test result database to the test nodes' hosts.

The time frame to be replayed can be selected as local time or SMT. The local time selection is to be preferred to the SMT one, as a mapping will be done from the SMT to find the corresponding local time. Especially in the following cases, selection of the time frame in SMT mode is not appropriate :

- The SMT was not running during the test or part of the test.
- The replay start SMT value anticipates any recorded data
- When nothing is to be replayed by one test node (can make sense in a distributed configuration).
- Usually in distributed configuration, as the time is used to synchronise the test nodes between themselves and best synchronisation is achieved with local time.

Speed factor

Another replay property which may be viewed/edited is a speed factor to be applied by the test nodes during replay. It may be set between 0.1 and 100, and will be sent to the test nodes during set–up.

If the speed factor should is set to 2 orhigher, it may happen that CGS is not able to process all data in time. Also the order of processing might not be guaranteed anymore, and synchronisation between test nodes can easily be lost.

Copyright per DIN 34

7.2.2.9.7 Test Configuration Application Control

The lowest section in the main window of TSCV represents the test system resources associated with the test configuration slot selected in the middle section.

One non–editable textfield displays the pathname of the currently selected test configuration.
Four non–editable text fields display a hierarchical CCU identifier. This is the identifier embedded in the slot's test configuration – which equals the CCU selected by the user in the process of loading from the mission database into the slot.

The scrolling list displays the node entries in the test configuration. When node entries are selected, properties of the first selected one may be viewed in a Properties window. When double clicking on an entry in the list, that entry will be viewed Properties window. The view is depending on whether the test configuration is **Idle** or **Active** it will open in edit or view mode. Actually, if the selected node is a CSS node, the window will always be in view mode. For an HCI, it is possible to specify whether or not the node shall participate, i.e. addressed for set–up. This also applies for a test node, and in addition it is possible to set test execution mode, a forced loading flag, and an MTP flag. When applying changes made for a node, this only affects the test configuration instance loaded into TSCV (which, however, will be saved in the test result database when creating a session), and TSCV will act upon them when setting up the slot. Only one participating test node may have the Master Test Processor (MTP) role. Thus, it will not be possible to set the MTP node non–participating, and the MTP role may only be switched to a participating node.

Please note that TSCV will ensure that it will not be possible to combine test node execution mode Normal and Replay in one slot.

The first column header in the node list is MTP, and one list entry representing a test node will contain a bracket in that column.

If TSCV determines that a node is not set–up as it should, e.g. that the Time Synchronization System is not responding, this will be reflected with an asterisk in the status column.

To find out more about the node status, the function **Check Status** should be invoked, either from the Configuration menu in the upper part of the main window, or from the Node menu above the node list. In the latter case, only those nodes selected in the list will be addressed.

Otherwise, the **Node** menu makes it possible to address individually selected nodes and give commands relating to these. The range of active menu options will depend on the state of the selected nodes, and the kind of node. Most options are only relevant for test nodes: Launch TES, Setup, Start, Suspend, Resume, Stop . The Launch Services option is for starting up all CGS services on the host indirectly designated by the node entry. The Shutdown option is used to shutdown HCI and TES applications, and the CGS services.

Copyright per DIN 34

7.2.2.10 TSCV Operation Control Procedures And Instructions

7.2.2.10.1 Interactive Mode

When TSCV is started in interactive mode, a graphical user interface is provided.

When TSCV executes in the interactive mode, all error conditions not handled internally will be escalated to the user, and resolved by him. No foreseen error condition shall cause the program to halt. Some errors may however not be resolved by the user of TSCV, but must be resolved by system maintenance, database maintenance etc.

If TSCV for some reason has not been able to perform user authorization (e.g. user has not system privilege "CGS:SETUP") TSCV will terminate (after prompting the user). The user account must be established – with sufficient authority to run TSCV.

The user may optionally specify command line parameters.
Command line parameters that can be used in interactive mode, are described in section 9.4.1.

7.2.2.10.2 Batch Mode

In batch mode, TSCV is started by a single command line. How TSCV shall perform its execution, is specified by the command line options.

When TSCV runs in batch mode, it will communicate a 'result' as a Unix completion status. It can be read by the program which invoked TSCV to determine the status upon termination. The exit codes are listed in section 9.5.2.

7.2.2.10.3 TSCV Housekeeping

TSCV uses a lock file to prevent that more than one instance is executing at any time. The file is named
tscv.lock
Note that the mere existence of the file does not prevent starting of TSCV. The file also must be regularly updated by a running TSCV.

TSCV uses files to maintain persistent data. Initially, when TSCV is installed, there will be no data files.

For each of the up to five loaded test configurations, TSCV will leave a file containing an image of the test configuration when terminating. The files will have names as
not_inuse_test_configurationx.dat
where x is an integer in the range of 1 to 5. When TSCV has executed a set–up, one of these files will be replaced with a file with a name as
cgs_test_configurationx.dat
where x is an integer in the range of 1 to 5.

When TSCV terminates, it will save information about the current TRDB status on a file named
tscv_saved_test_session_data
This file will be consulted during start–up, and warnings will be issued if it is found that the test session which was open when TSCV last terminated is no longer the one which is open.

When TSCV terminates, it will store information about the state of the testnodes on a file named
tscv_saved_testnode_status.dat
This file will be consulted during start–up, and warnings will be issued if it is found that any of the testnodes reports to be in another state than what was the case when TSCV terminated.

Copyright per DIN 34

7.2.2.11 TSCV Reference Information

In the following, the windows and user interface functions of TSCV are described and reference information is given for each.

7.2.2.11.1 Help Method

Help is provided for each window component (panel item) of the TSCV dialogues. To get help, the cursor must be positioned at the window component (panel item) for which help is requested. Then the keyboard button labelled **Help** is pushed. The help text will be displayed in a separate window.

To obtain a help summary for the window, the cursor must be positioned inside the window, but not at one of the window components.

If the user want to attach help to another key than the default key for help, the help key may be specified by setting the resource *OpenWindow.KeyboardCommand.Help*.

7.2.2.11.2 Screen Definitions and Operations

This section describes the functionality of the user interface provided when TSCV is invoked in interactive mode. The content of windows, its components and what to be accomplished by the windows' controls, are described.
The section is the basis for the on–line help texts. Therefore, the description has been adapted to such a usage. Figures of TSCV dialogues will not be part of the on–line help output.

Copyright per DIN 34

7.2.3 The TSCV Main Window

7.2.3.1 General

When TSCV is started in interactive mode, its **'Main Window'** will be opened. The window is divided into three parts (panels), one for the test site, one for test configurations and one to handle the nodes of the currently selected test configuration.

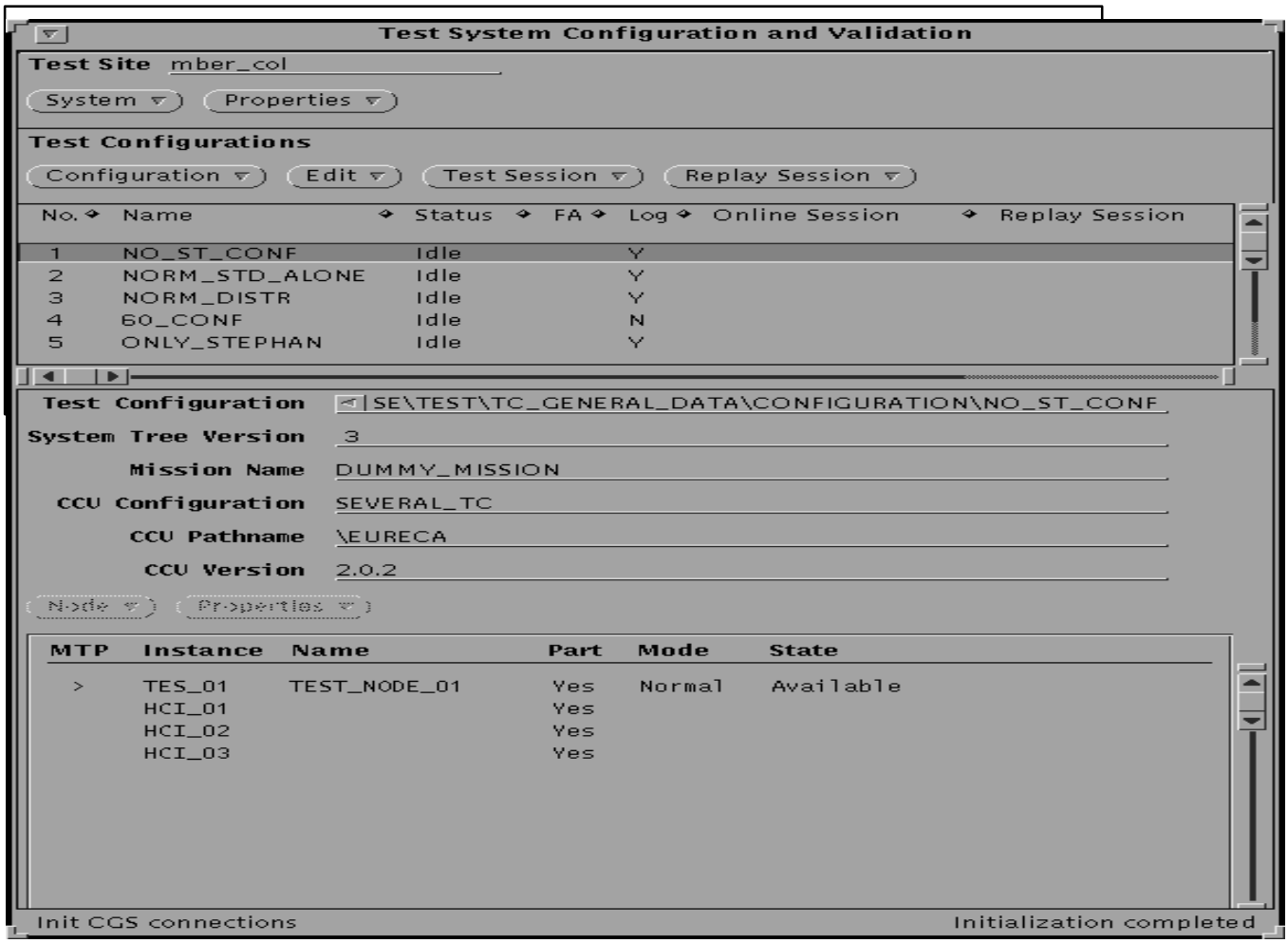


Figure 7–5 *: The TSCV Main Window*

The **'Main Window'** contains a set of menus used to control the application, and some panel items showing TSCV data and CGS status. Subwindows are opened when the user activates menu items where the label ends with three dots. The subwindows are in general unique for the function associated with the menu item that brings them up. Some exceptions do however exist:

- The **'Select CCU'** window
  The window might be opened in two contexts, one when specifying a CCU as a part of loading a test configuration, and one when specifying the selection criteria filter to list test sessions in the *'Multi Purpose Test Session'* window.

Copyright per DIN 34

- The **'Multi Purpose Test Session'** window
  To handle selection of test session, one window is used in three different contexts, with a different title an a slightly different content. The three contexts are:
  – Maintain test session, to close sessions, delete sessions and delete from the default session.
  – Load a test configuration from TRDB for replay
  – Assign a test session for replay

- The **'Display Request Window'**
  The window is opened in various contexts when the user requests to view data.

TSCV does not allow parallel execution of operations, i.e. the user can not activate a new operation before the previous has completed. TSCV will use a "busy pointer" to indicate that an operation is executing.

Many of the operations that the user can activate from the **'Main Window',** or its subwindows, only apply when TSCV is in a certain state. When the state of TSCV changes, TSCV will set the appropriate panel items active or inactive (dimmed) to give the user a visual feedback of operations which are enabled or disabled. A dimmed panel item means that the functionality connected to the item is not available in the current context.

- The **Test Site** panel
  The panel is related to the test system. It contains the two menus used to control the test system, in addition to a text field displaying the current test site.
  The **'System'** menu, contains the options to launch CGS services and to shutdown the test system.
  The **'Properties'** menu contains the options to configure and view test system data, i.e to open windows in order to maintain the *'System Topology Table'* and to view *'Software Versions'*.

- The **Test Configurations** panel
  The panel is related to loaded test configurations. It displays a list of the loaded test configurations up to the maximum of five, and the menus needed to control the currently selected entry in the list of test configurations.
  The **'Configuration'** menu contains the options to check status, setup, start test, suspend test, resume test, stop test and to shutdown a test configuration.
  The **'Edit'** menu contains the options to load a test configuration from MDB, to load a test configuration from TRDB for replay, to unload a test configuration and to view test configuration data.
  The **'Test Session'** menu contains the options to create, close and delete a test session.
  The **'Replay Session'** menu contains the options to assign for replay a test session to a test configuration loaded from MDB and to open the *'Replay Properties'* window

- The **Selected Test Configuration** panel
  The panel is related to the selected slot in the list of test configurations. For empty slots, nothing is displayed. When a test configuration is selected, its pathname is displayed and

Copyright per DIN 34

its CCU identifier is partly displayed. The nodes associated to the test configuration is displayed in the list of nodes. When doubleclicking one entry in the list, the associated attributes are dis0polayed in the *'Node Property Sheet'* window.
The menu choices found in the **'Selected Test Configuration'** panel affects only the *selected* nodes in the list of nodes. Two menus are provided:
The **'Node'** menu contains the options to check status, launch services, launch TES, set-up, start test, suspend test, resume test, stop test and to shutdown selected nodes.
The **'Properties'** menu contain one menu item, **'Node...'** , which displays the attributes of the first selected node in the **'Node Property Sheet'** window.

7.2.3.2 The 'System' Menu

The menu contains options to control the test system.

7.2.3.3 The 'System–>Launch Services' Menu Option

The menu option is used to launch the CGS services. It is possible to start the CGS services at any time as long as TSCV is not requested to prevent other user activities. It will not do any harm on applications already running.

7.2.3.4 The 'System–>Shutdown' Menu Option

The menu option is used to shutdown the test system. It is possible to perform a shutdown of the test system at any time as long as TSCV is not requested to prevent other user activities.

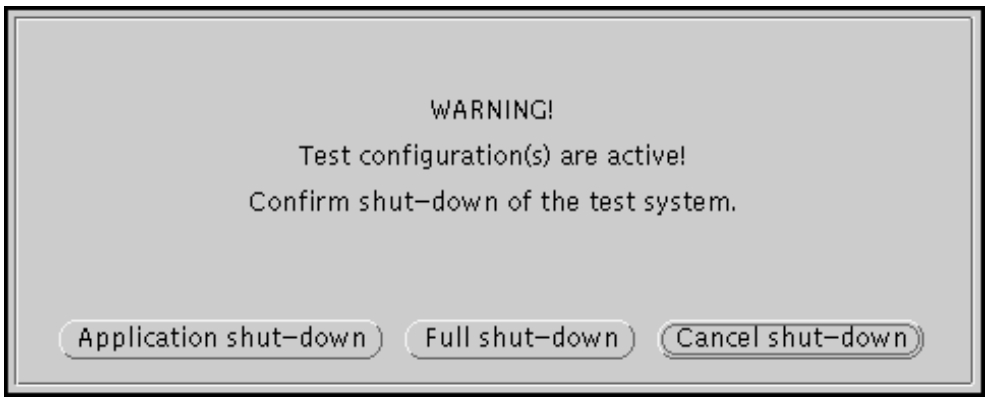


Figure 7–6 *: Confirm Shutdown*

Before actually executing the function, the user will be asked to confirm the shutdown request.

If the request is answered by **'Application Shutdown'**, test configurations which have been set–up (is in the state **'Active'**), will be stopped. Stopping a test configuration implies to stop test execution on participating testnodes and close the (possibly) open test execution session.

Then all test and workstation nodes appearing in the *'System Topology Table'* will be commanded to shutdown.

Finally, the *'Central DBS'* will be commanded to shutdown.

Copyright per DIN 34

If the request is answered by **'Full Shutdown'**, in addition to application shutdown, the remaining CGS applications, e.g. *Network Software* and *TSS*, will be shut down.

If the request is answered by **'Cancel Shutdown'**, shutdown will not be performed.

7.2.3.5 The 'Properties' Menu

The menu contains options to maintain the system topology and to view software versions.

7.2.3.6 The 'Properties–>System Topology...' Menu Option

The menu option is used to open the *'Maintain System Topology'* window, which may be used to view, add, change and remove entries in the *'System Topology Table'*.

7.2.3.7 The 'Properties–>Software Versions...' Menu Option

The menu option is used to view the software versions in a display request window.

7.2.3.8 The 'Configuration' Menu

The menu contains options to control the test configuration currently selected in the list of test configurations.

7.2.3.9 The 'Configuration–>Check Status' Menu Option

The menu option is used to check the status of test nodes and DBS. TSCV prompts the testnodes for their status and renders it in the node list.
When the TSP status inquiry reports an error, the user gets the options to set–up the TSP or to continue. If setup is selected, implicit launch will be performed if the TSP is not responding. Note that TSP status are checked only for active test configurations.

Additionally a window pops up and displays information about the NTP and SMT status of the participating hosts:

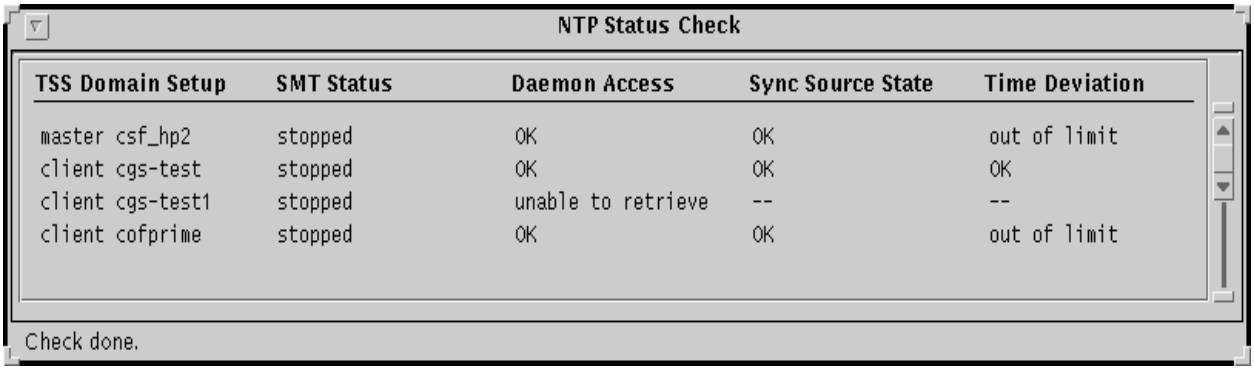


Figure 7–7 *: The NTP Status Check Window*

First column "TSS Domain Setup":

Shows the participating master and clients of TSS Domain (role + hostname)

Copyright per DIN 34

Second column ”SMT Status”:

The SMT status of the corresponding host. It can be ’running’ or ’stopped’.

Third column ”Daemon Access”:

It can be ’OK’ or ’unable to retrieve’.
If it is ’unable to retrieve’ verify that the xntp–daemon is running (ps –ef | grep xntpd). If the process is running look for error messages in CGS–Startup outputs.

Fourth column ”Sync Source State”:
It can be ’OK’, ’master unreliable’ or ’—’ (undefined).
If it is ’master unreliable’ the host cannot reach the xntp–daemon on its master host or the master host’s NTP time information are unreliable. If the host is domain master and has no external time source, then there could be problems with the internal hardware clock. If the host ist domain master and has a external time source, then the time source information are unreachable or unreliable.

Fifth column ”Time Deviation”:
It can be ’OK’, ’out of limit’ or ’—’ (undefined).
If it is ’out of limit’ NTP synchronization is running without problems, but the time difference between master and client is greater than allowed.
Reasons for occurence of this status can be a initial installation, configuration changes concerning master–client assignments or a time jump on master side.
In case of initial installation or configuration changes it can take up to 48 hours to sync.

The **’Check Status’** option, also includes check of DBS . If the check concludes that DBS is down, it will be launched.

7.2.3.10 The ’Configuration–>Setup’ Menu Option

The menu option is used to set–up a test configuration.

TSCV’s actions are determined from the contents of the test configuration currently selected in the list of test configurations.

Initially, TSCV will check that there is at least one participating testnode, i.e. that the *’participating flag’* is set for at least one of the testnode entries. Then it will be ensured that the central DBS is up and running (invoking the command to start, if necessary).

At this stage, the status of the test configuration is changed. In the main window, this is recognized by a change in the **’Status’** field in the test configuration list.

Before activating set–up, the user may have associated a test session name with the test configuration (see menu option **’Sessions–>Create’**). If so, TSCV will create a session with that name in the database. If no such session is defined, the default test session will be used. Note that the default test session can only be used by one active test configuration. Set–up will be denied when the default test session already is allocated to an active test configuration.

Next step is to ensure that the TES process on all participating testnodes is up and running (invoking the command to start, if necessary). Then SMT/TSS initialization is performed.

Now the CCU in MDB associated with the test configuration will be set as the global default CCU for the test configuration.

Copyright per DIN 34

Then the turn has come to set–up the testnodes. This implies to command the TES process on the individual testnodes to execute initiation, i.e. to load configuration data from MDB and enter the execution mode **'Normal'** or **'Simulation'** (as specified in the test configuration).

Finally, the testnodes will be synchronized. If there are more than one participating testnode, TSCV will inform TES on each of the testnodes which other nodes are participating. The TES instances will then exchange configuration information necessary for their cooperation.

The set–up is considered to be executed when TSCV has started setting up the testnodes. If any problems are encountered before that, the function is abandoned and the state of the test configuration remains **'Idle'**. Regardless of any problem occurring during set–up of the testnodes, the result is that the test configuration is in the state **'Active**.

During the setup sequence, progress is reported through a dedicated progress report/abort window. The window contains one button labelled **'Abort'**. Aborting the setup sequence requires a confirmation from the user. When confirmed, the setup sequence will be aborted after end of the operation performing the current setup step.

7.2.3.11 The 'Configuration–>Start' Menu Option

The menu option is used to start test execution on a test configuration.

If this option is selected for a test configuration in state **'Idle'**, the set–up function as described for the **'Configuration–>Setup'** menu option will be executed, followed by commanding all participating testnodes to start test execution.

This menu option will also be selectable when the selected test configuration is in the state **'Active'**, and at least one of the testnodes is not in the state **'Executing'**. In this case, each participating and non–executing testnode will be commanded to start test execution.

7.2.3.12 The 'Configuration–>Suspend' Menu Option

The menu option is used to suspend testing for test nodes in simulation or replay mode. The menu option will be available for test configurations in state **'Active'** provided that there are executing node(s) of mode **'Sim'** or **'Replay'**. The test execution for all executing test nodes of mode **'Sim'** and **'Replay'**, will be suspended (enters status **'Suspended'**).

7.2.3.13 The 'Configuration–>Resume' Menu Option

The menu option is used to resume testing for suspended nodes. The menu option will be available for test configurations in state **'Active'** provided there are node(s) with status **'Suspended'**. All suspended nodes will be resumed, i.e. enter the status **'Executing'**.

7.2.3.14 The 'Configuration–>Stop' Menu Option

The menu option is used to *stop* an executing test configuration.

Any testnode which executes testing (status **'Executing'**) will be commanded to stop testing (enter the status **'Available'**).

If there is an open test execution session in the database, this will be closed.

7.2.3.15 The 'Configuration–>Shutdown' Menu Option

The menu option is used to shutdown a test configuration, i.e. shutting down all participating nodes of the test configuration. Before actually executing the function, the user will first be asked to confirm the shutdown request.

Copyright per DIN 34

If the request is answered by **'Application Shutdown'**, and the test configuration is in state **'Idle'**, then all participating test and workstation node are shutdown. If the test configuration is in state **'Active'**, then test execution is stopped on the participating test nodes before the tes tnodes and workstation nodes are shutdown.

If the request is answered by **'Full Shutdown'**, in addition to application shutdown, the CGS services, e.g. Network Software and TSS, will be shut down.

A test configuration shutdown will never change the state of the test configuration.

If the request is answered by **'Cancel Shutdown'**, shutdown will not be performed.

7.2.3.16 The 'Edit' Menu

The menu contains options to load, unload and view test configurations.

7.2.3.17 The 'Edit–>Load for online test...' Menu Option

When selected, the menu option opens the **'***Load Test Configuration for Online Test'* subwindow, which is used to load a test configuration from MDB into an empty slot or a slot containing an **'Idle'** test configuration.

7.2.3.18 The 'Edit–>Load for Replay...' Menu Option

When selected, the menu option opens the **'***Load Replay Session'* subwindow, which is used to load a test configuration for replay from TRDB into an empty slot or a slot containing an **'Idle'** test configuration.

7.2.3.19 The 'Edit–>Unload' Menu Option

The menu option is used to unload an **'Idle'** test configurations. The test configuration slot will be marked as empty.

If any attributes of the test configuration to be unloaded has changed, a warning is issued and the user has to confirm the unload request.

Figure 7–8 *: Confirm Unload*

7.2.3.19.1 The 'Edit–>View...' Menu Option

When the menu option is selected, TSCV will format a textual representation of the attributes associated with the currently selected test configuration in the list of test configurations. The result will be displayed in a

Copyright per DIN 34

text window.
Also when double clicking in the list of test configurations, the selected test configuration will be viewed in a text window.

7.2.3.20 The 'Test Session' Menu

The menu contains options to create, close and delete test sessions.

7.2.3.21 The 'Test Session–>Create' Menu Option

When the menu option is selected, the session creation window will be opened. The created session will be associated with the test configuration currently selected in the list of test configurations.

The menu option will be dimmed if the database already contains an open test session for the selected test configuration.

7.2.3.22 The 'Test Session–>Close' Menu Option

When the menu option is selected, TSCV will close the session associated with the selected test configuration. A window pops up where the status (percent of saved files) of closing is displayed. If the session is closed the window disappears. The user has the option to abort the waiting for closing the test session. Therefore he can press the 'Abort waiting' button. (Abort waiting do not mean abort closing the session!) If the test configuration does not have an open session, the menu option will be dimmed.

Figure 7–9 : *Close Session*

7.2.3.23 The 'Test Session–>Maintain' Menu Option

When the menu option is selected, the *'Maintain Test Session'* subwindow will be opened.

7.2.3.24 The 'Replay Session' Menu

The menu contains options to assign test sessions to be replayed and to open a window to view and edit replay properties.

7.2.3.25 The 'Replay Session–>Assign...' Menu Option

When the menu option is selected, the *'Assign Replay Session to Test Configuration'* window is opened.

Copyright per DIN 34

7.2.3.26 The 'Replay Session–>Properties...' Menu Option

When the menu option is selected, the *'Replay Properties '* window is opened. That window is used to view and edit properties of replay test configurations

7.2.3.27 The Test Configuration List

TSCV can 'manage' up to 5 loaded test configurations. The list shows the test configurations identifier, status and session information. The list entry which is selected represents the test configuration currently 'in focus'.

TSCV maintain a persistent copy of its database; thus, after terminating and restarting TSCV at a later occasion, the test configuration list will again render the same information.

Double clicking in the list will have the same effect as selecting the **'Edit–>View...'** menu choice. TSCV will format a textual representation of the attributes associated with the selected test configuration. The result will be displayed in a text window.

The list of test configurations consist of five columns:

*The **'No.'** column*

This column identifies the test configuration slot number in the range 1 to 5.

*The **'Name'** column*

This column identifies the name of the test configuration, represented by the last part of its path name. When the slot does not contain a test configuration, the **'Name'** column is rendered **'None'.**

*The **'Status'** column*

The **'Status'** of the loaded Test Configurations can be **'Idle'** or **'Active'**. Initially, when a test configuration is loaded (read from the mission database), it is **'Idle'**. If the user later commands TSCV to execute set–up, the selected test configuration entry becomes **'Active'**, and will then represent an image of a group of participating, cooperating nodes (workstations and testnodes).

*The **'FA'** column*
Indicator of active final archiving

*The **'LOG'** column*
Indicator of active logging

*The **'Online Session'** column*

An **'Online Session** 'may be associated with each of the loaded test configurations. While the testnodes within an executing test configuration are processing, testdata are generated. These are stored in the *'Test Result Database'* within an *'envelope'* called a **test session**. The user may associate a session name with a test configuration when it is **'Idle'** as well when it is **'Active'**.

*The **'Replay Session'** column*

A **'Replay Session**' may be associated with a test configurations when the recorded data shall be replayed by the loaded test configuration. The test configuration may either have been loaded from MDB or loaded from TRDB (embedded in the test session data).

7.2.3.28 The 'Test Configuration' Text Field

The text field display the path of the currently selected test configuration.

Copyright per DIN 34

7.2.3.29 The 'System Tree Version' Text Field

The textfield displays the **'System Tree Version'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

7.2.3.30 The 'Mission Name' Text Field

TBD

7.2.3.31 The 'CCU Configuration' Text Field

The text field displays the **'CCU Configuration'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

7.2.3.32 The 'CCU Pathname' Text Field

The text field displays the **'CCU Pathname'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

7.2.3.32.1 The 'CCU Version' Text Field

The text field displays the **'CCU Version'** part of the CCU identifier of the selected test configuration. For an empty slot, the text field will be empty.

7.2.3.33 The 'Node' Menu

The menu contains options to control selected nodes.

7.2.3.34 The 'Node–>Check Status' Menu Option

The menu option is used to update the values of the column **'State'** for test nodes selected in the node list. Hence, the operations initiated will be equal to the operations associated with the **'Configuration–>Check'** menu choice, except that the **'Node–>Check'** menu choice addresses only selected nodes of the test configuration. For initialized nodes, the column **'Mode'** will also be updated according to the acquired status.

The menu option will be dimmed when one or more of the selected nodes are in use by another active test configuration, i.e. the status field is rendered **'In use'**.

This menu option is the only option available if the status of the selected test node is **'Unknown'**.

7.2.3.35 The 'Node–>Launch Services' Menu Option

By this menu option, the user may initiate the start of the CGS services on the selected nodes.

The menu option will be dimmed if the state of the test node is **'Unknown'**. In this case **'Check Status'** has to be performed before further operations are allowed.

7.2.3.36 The 'Node–>Launch TES' Menu Option

By this menu option, the user may initiate the launching of the TES process on a node.

Copyright per DIN 34

The option will be selectable only when the **'State'**, of each entry selected in the node list, are **'No contact'**.

7.2.3.37 The 'Node–>Setup' Menu Option

The menu option is used to execute set–up on individual test nodes, i.e. to command selected testnodes to load configuration data from the mission database. If the test configuration is in state **'Idle'**, the state will be changed to **'Active'**.

A participating test node will change state from **'Available'** to **'Idle'** when the test node has been commanded to initialize.

This function would be used for instance if one of the testnodes failed to complete its set–up successfully while TSCV executed the set–up function on test configuration level. Another example when this function could be useful was after some kind of crash on one of the testnodes; it would then be possible to launch and subsequentially set–up TES on that testnode only, without performing set–up for the whole test configuration.

This option will be dimmed when one of the selected test nodes have state **'Unknown', 'In use'**, **'Executing'** or **'Error'**.

7.2.3.38 The 'Node–>Start' Menu Option

This menu option is used to command start testing on individual testnode level. If the test configuration is in state **'Idle'**, setup of the whole test configuration will be performed implicitly.

This option will be dimmed when one of the selected test nodes have state **'Unknown', 'In use'**, **'Executing'** or **'Error'**.

7.2.3.39 The 'Node–>Suspend' Menu Option

The function associated with this menu option is to command the selected test node(s) to suspend test execution.

This option will only be active when each of the selected test nodes are in the mode **'Simulation'** and the status is **'Executing'**.

7.2.3.40 The 'Node–>Resume' Menu Option

The menu option is used to command the selected test node(s) to resume test execution.

This option will only be active when each of the selected test nodes are in the mode **'Simulation'** and the status is **'Suspended'**.

7.2.3.41 The 'Node–>Stop' Menu Option

The menu option is used to command stop test execution on individual testnode level.

This option will only be active if each of the selected testnodes is in the state **'Executing'**.

7.2.3.42 The 'Node–>Shutdown' Menu Option

With this menu option, it is possible to command individually selected test and work–station nodes to shutdown. It is also possible to shutdown the CGS services on the selected nodes.

Copyright per DIN 34

The menu option will be dimmed if the state of the test node is **'Unknown'**.

7.2.3.43 The 'Properties' Menu

The menu option is used to open the *'Node Property Sheet'* window.

7.2.3.44 The 'Properties–>Node...' Menu Option

This menu option is used to open the 'Node Property Sheet' window. When the node list contains multiple selections, then the attributes of the first selected entry will be displayed. A double click at another entry in the list, will display attributes of the double clicked entry.

If the status of the selected test configuration is **'Idle'**, it will be possible to change the value of some of the attributes.

If the status of the selected test configuration is **'Active'**, the window will be 'read only'.

7.2.3.45 The List of Nodes

The test nodes, work station nodes and the simulation node of the test configuration in focus will be listed here. Please note that when the status of the test configuration is **'Idle'**, then **all** node entries in the test configuration are rendered, while only the **participating** nodes appear when the test configuration is **'Active'**.

As long as the test configuration is **'Idle'**, the **'Node Properties Sheet'** window can be used to edit a subset of attributes for nodes displayed in the node list.

When double clicking in the list, the node property sheet for the selected node is displayed.

The list is formatted in columns with headers **MTP, Instance, Name, Part, Mode and State:**

*The **'MTP'** column*
When more than one testnode is participating, one must have the role **'Master Test Processor'**. Under the column header **'MTP',** this is indicated by a **'>'** sign in front of the testnode name.

*The **'Instance'** column*
The physical node name is showed in the column **'Instance'**.

*The **'Name'** column*
The logical node name, i.e. the last part of the pathname, is displayed in the column **'Name'.**

*The **'Part'** column*
Whether or not the node is/will be participating is indicated by a **'Yes'** or a **'No'** in the column headed by **Part**. When the test configuration is executing, only the participating nodes are rendered, hence only **'Yes'** will be seen.

*The **'Mode'** column*
The execution mode associated with the testnode entries in the test configuration is displayed in the column with header **'Mode'**. This mode value will be applied when setting up the test configuration, as well as when setting up individual testnodes. For initiated TES instances (status **Idle**, **Executing** or **Suspended**), the **'Mode'** column renders real mode. Else the **'Mode'** column renders the desired mode as defined/displayed in the node property sheet.

The different mode options possible are as follows:

Copyright per DIN 34

Normal – This is the normal operational mode where TES is interacting with Special Application Software (SAS).
Sim – In this mode TES does not interact with the Special Application Software. Instead, TES simulates this interaction.
Replay – In this mode TES replays a previous recorded session. Instead of interacting with the Special Application Software, TES ”sends” and ”receives” the exactly same data as it did when the session was recorded.

*The **’State’** column*
The node state as known by TSCV is rendered in the corresponding column. The status is updated as response to user commands; either when issuing a command to set–up, start or stop the test configuration, to set–up, start, suspend, resume or stop individually selected testnodes, or when the **’Check Status’** function is executed.

The different status options possible for test nodes are as follows:

Unknown – The TES instance has never been commanded by TSCV.
No contact – TSCV has no contact with the TES instance.
Error – The TES instance is in an error state.
Available – The TES instance is available and may commanded.
In use – The TES instance is in use by another active test configuration. The instance may not be commanded as part of this test configuration.
Idle – The TES instance has been initialized and is ready to start test execution.
Executing – The TES instance is executing a test.
Suspended – The TES instance has been suspended.

An asterix is appended to the TES statues *’Idle’*, *’Executing’* and *’Suspended’* if synchronization or the Time Service has not been set–up correctly on the node.

For the other instance in the node list, i.e. work–station and simulation nodes, the only possible value shown in the status column is the asterix. This value indicates that the Time Service has not been set–up correctly on the node hosting the instance.

7.2.4 The Load Test Configuration Window

7.2.4.1 General

The **’Load Test Configuration for Online Test’** window will be opened when the main window menu choice **’Edit–>Load for Online Test...’** is activated. The window is used to load a test configuration end item from the mission database. TSCV has space (’slots’) for up to 5 test configurations, each with a corresponding entry in the test configuration list in the main window. When the **’Load’** button is activated in this sub–window, the selected test configuration will be loaded into the slot currently selected in the main window. Loading a test configuration can also be initiated by double clicking in the list of test configurations.

Copyright per DIN 34

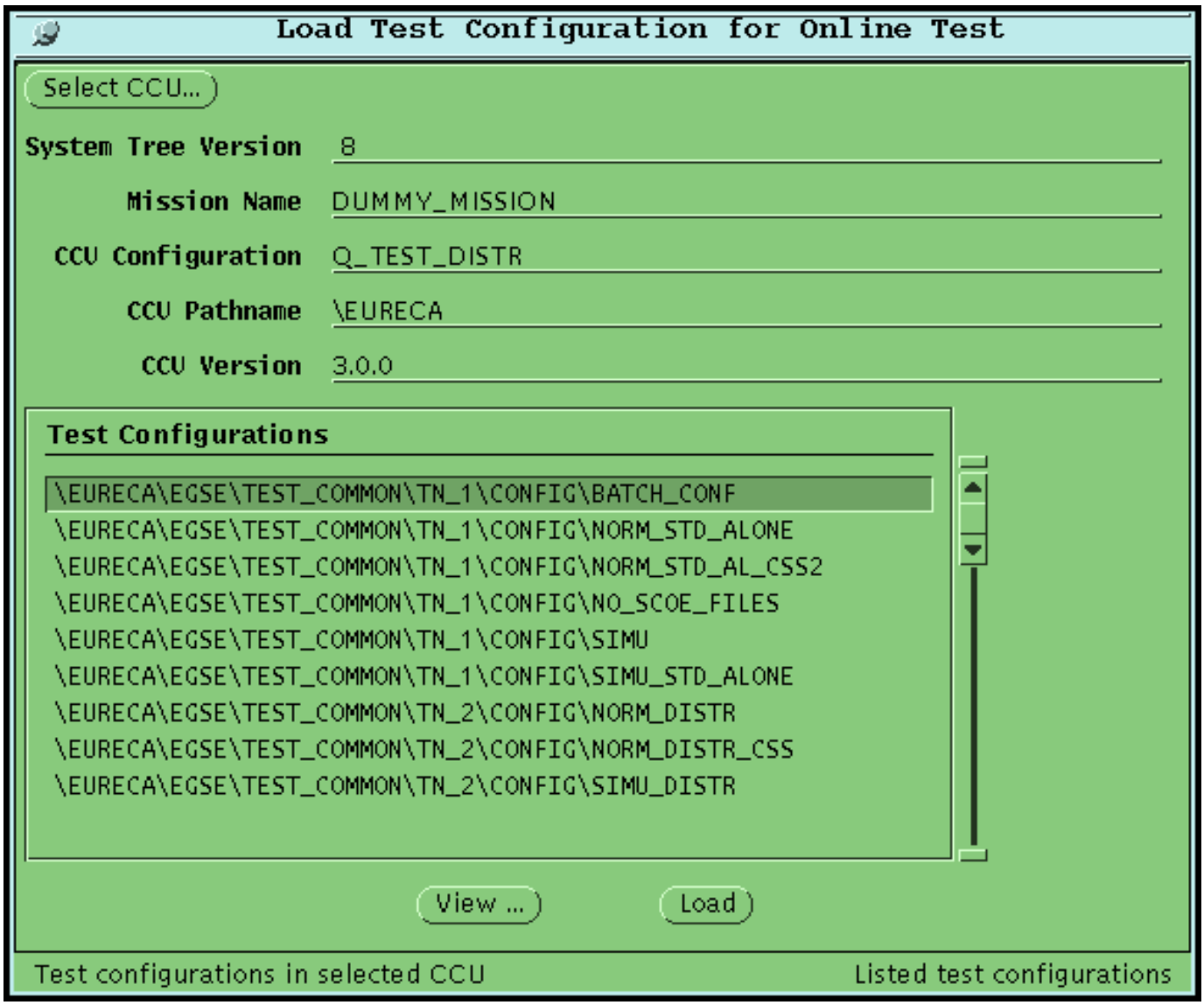


Figure 7–10 *: The Load Test Configuration for Online Test Window.*

The window will list all the test configuration end items in the mission database that ’belongs’ to the CCU viewed in the associated text fields. When the window is opened, the content of the CCU text fields are dependant on the current selection in the list of test configurations of the TSCV main window. If an idle test configuration is currently selected, the associated CCU will be rendered into the text fields. If the selected slot is empty, the CCU will be the one associated with the last selected test configuration.

If the CCU text fields are empty when the window is opened, it means that TSCV was not able to work out which test configuration that was the last selected one. Note that initially, when TSCV is taken into use in a new environment, all slots will be empty; hence the CCU text fields will also be blank.

The button **‘Select CCU ...’** can be used to activate another sub–window for CCU selection. After selection, the list of test configurations will be updated to contain the test configurations that ’belongs’ to the selected CCU.

Copyright per DIN 34

The window lets you view the contents of the test configurations in a text window by selecting the **'View...'** button.

When TSCV has no record of the test configuration last selected, e.g. after the very first invocation of TSCV, then the CCU text fields and the buttons **'View ...'** and **'Load'** will be dimmed. In that case, the user has to activate another sub–window for CCU selection before a test configuration can be selected for loading.

7.2.4.2 The 'Select CCU...' Button

When the **'Select CCU ...**' button is activated, another window opens, in which a CCU can be selected. The CCU thus selected will be rendered into the associated text fields**.** The list of test configurations will be updated to contain the test configurations that 'belongs' to the selected CCU.

7.2.4.3 The 'System Tree Version' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'System Tree Version'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

7.2.4.4 The 'CCU Configuration' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Configuration'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

7.2.4.5 The 'CCU Pathname' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Pathname'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

7.2.4.6 The 'CCU Version' Text Field

The list of test configurations displays mission database end–items that 'belongs' to a CCU. The text field displays the **'CCU Version'** part of that CCU identifier.
The text field will be empty and dimmed when a CCU is not identified by TSCV.

7.2.4.7 The 'Test Configurations' List

The list contains the pathname of all test configurations associated with the selected CCU (displayed in the CCU text fields). Whenever a CCU is rendered into the CCU text fields – either when the window is opened or when a CCU has been selected in the CCU subwindow – TSCV will retrieve all the associated test configurations from the mission database and display their pathname identifier in this list.

When double clicking in the list of test configurations, loading of the test configuration will be initiated.

7.2.4.8 The 'Load' Button

When the **'Load**' button is activated, TSCV will load the database end–item identified by the currently selected test configuration in the **'Test Configurations'** list. There is space (slots) for up to 5 loaded test con-

Copyright per DIN 34

figurations, and the retrieved end item will go into the slot currently selected in the main window. When the loading is completed, this window will be closed, and the main window will be updated to reflect the new test configuration.

If the '**Load'** button is dimmed, this means that no CCU is identified. A CCU has to be selected by activating the button '**Select CCU ...'**

7.2.4.9 The 'View...' Button

When the **'View ...**' button is activated, test configuration attributes associated with the current selection in the **'Test Configurations'** list, will be formatted for display purposes. A dedicated text window is used to view the attributes.

If the '**View ...'** button is dimmed, it means that no CCU is identified. A CCU has to be selected by activating the button '**Select CCU ...'**

7.2.5 The Select CCU Window

7.2.5.1 General

The **'Select CCU'** window can be used in two contexts:

- **Loading Test Configuration** window:
  The *'Select CCU'* window is opened by pushing the **'Select CCU ...'** button in the *'Load Test Configuration***'** window. It is opened to specify a CCU identifier in the mission database. The selected CCU will be used to retrieve associated test configurations.
  If the *'Load Test Configuration'* window already has a current CCU selection, it will be used as the default selection when the *'Select CCU'* window is opened.
  When a CCU is identified, pushing the **'Apply'** button transfers the CCU into the CCU text field in the *'Load Test Configuration'* window as the 'current CCU'.
  Note that if the **'Select ...'** button in the *'Multi Purpose Test Sessions'* window is pushed, the context of the *'Select CCU'* window will change accordingly
- **Multi Purpose Test Sessions** window:
  The *'Select CCU'* window is opened by pushing the **'Select CCU ...'** button in the *'Multi Purpose Test Sessions'* window. It is opened to select/specify a CCU identifier in the mission database. The selected CCU will be used as filter when a list of test sessions is generated (only associated test sessions will be listed).
  If the *'Multi Purpose Test Session'* window already has a current CCU selection, it will be used as the default selection when the *'Select CCU'* window is opened.
  When a CCU is identified, pushing the **'Apply'** button transfers the CCU into the CCU text fields in the *'Maintain Test Sessions'* window as the 'current CCU'.
  Note that if the **'Select CCU...'** button in the *'Load Test Configuration for Online Test'* window is pushed, the context of the *'Select CCU'* window will change accordingly

The **'Select CCU'** window consists of two main parts:

Selection of components in the CCU hierarchy
The selection of a CCU is based on a hierarchical approach. List buttons are used to display a list of components of the corresponding level in the hierarchy. The component collection listed, is based on the selections

Copyright per DIN 34

made for the higher levels in the hierarchy.
The selection of the CCU component at one level in the hierarchy, is done by selecting an item displayed in the list. The list button for the next level in the hierarchy, is made available. By pushing that button, the CCU components based on the current selection, are displayed in the list.
Double clicking in the list, has the same effect as first selecting an item, and then pushing the button to display the next level in the hierarchy.

Current Selection
For each of the CCU components, there is one corresponding text field, where the current selection is displayed. When an item in the currently displayed list is selected, the corresponding text field is updated. Naturally, if one goes back up in hierarchy, the lower level fields will be blanked and dimmed.
The text fields labelled **'CCU Configuration Name'** and **'CCU Pathname'** do not correspond directly to any of the **List** buttons; they are associated with the button **'CCU'**.

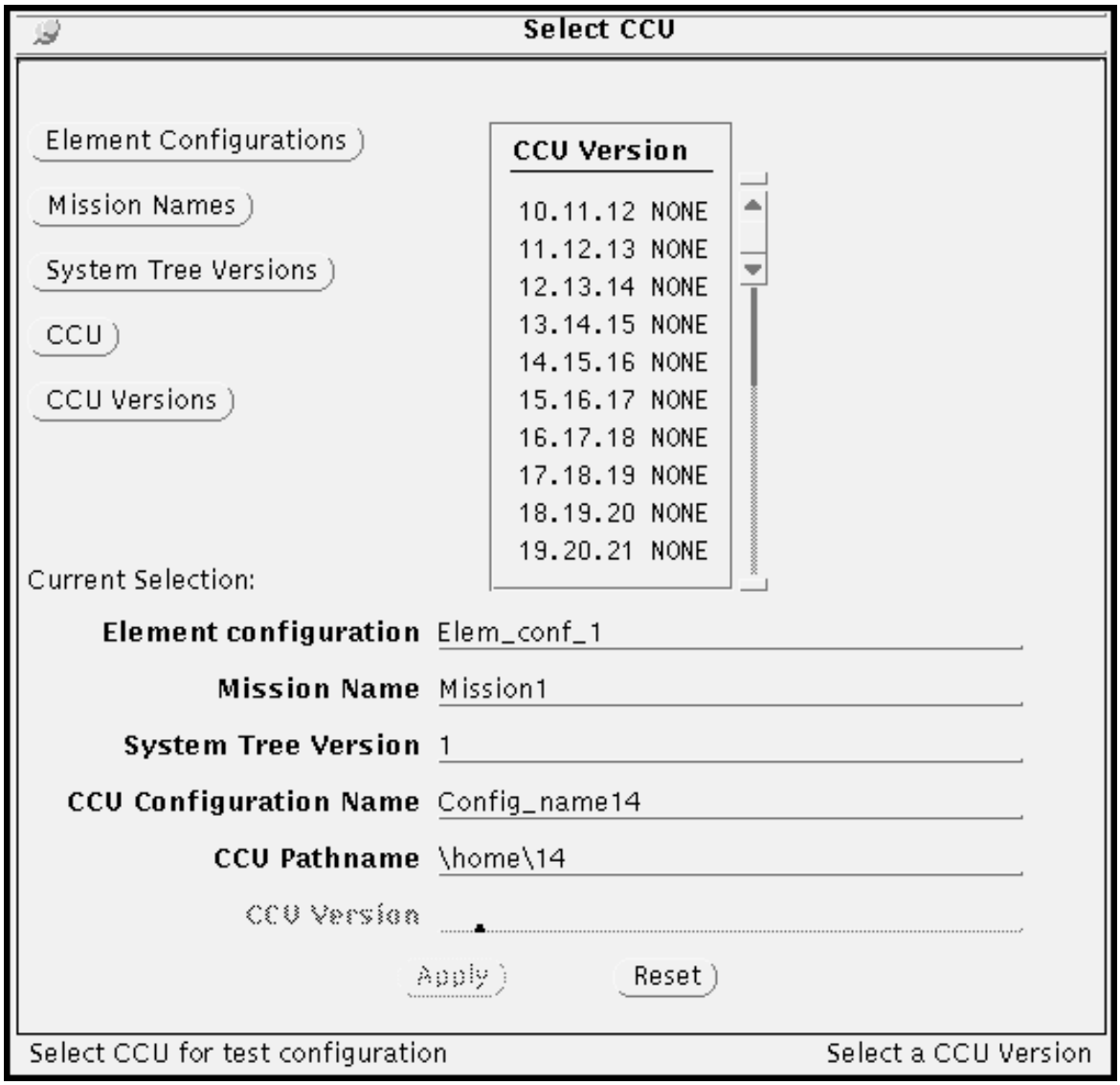


Figure 7–11 : *The Select CCU Window*

Copyright per DIN 34

7.2.5.2 The 'Element Configurations' Button

When this button is pushed, the list will display available element configurations. All other buttons, except the **'Reset'** button, will become inactive (dimmed). When selecting one of the items in the list, the **'Mission Names'** button will become active and the selected list item is displayed in the **'Element configuration'** text item for current selections. Double clicking in the list of element configurations will set the double clicked item as selected, list the associated mission names and enable the **'Mission Names'** button.
Note that any selections made, will be lost when pushing the '**Element Configurations'** button.

7.2.5.3 The 'Mission Names' Button

The button is available when an element configuration has been selected, i.e. displayed in the **'Element configuration'** text item for current selections. When the button is pushed, the mission names based on the selected element configuration are displayed in the list. By selecting an item in the list, the **'System Tree Versions'** button will become active and the selected list item is displayed in the **'Mission Name'** text item for current selections. Double clicking in the list of mission names will set the double clicked item as selected, list the associated system tree versions and enable the **'System Tree versions'** button
Note that when pushing the **'Mission Names'** button, all selections other than the element configuration selection, will be lost.

7.2.5.4 The 'System Tree Versions' Button

The button is available when a mission name has been selected, i.e. displayed in the **'Mission Name'** text item for current selections. When the button is pushed, the system tree versions based on the selected element configuration and mission name, are displayed in the list. By selecting an item in the list, the **'CCU nodes'** button will become active and the selected list item is displayed in the **'System Tree Version'** text item for current selections. Double clicking in the list of system tree versions will set the double clicked item as selected, list the associated CCU configuration names and enable the **'CCU Nodes'** button.
Note that when pushing the **'System Tree versions'** button, all selections other than the element configuration and mission name selections, will be lost.

7.2.5.5 The 'CCU' Button

The button is available when a system tree version has been selected, i.e. displayed in the **'System Tree Version'** text item for current selections.

When the button is pushed, the CCU configuration names based on the selected element configuration, mission name and system tree version, are displayed in the list. Note that the CCU configuration names not necessarily are unique. By selecting an item in the list, the **'CCU Versions'** button will become active and both the text items **'CCU Configuration Name'** and the **'CCU Pathname'** for current selections, will be filled in. Double clicking in the list of CCU nodes will set the double clicked item as selected, list the associated CCU versions and enable the **'CCU Versions'** button. When this button is activated, the scrolling list will display the CCU configuration names. By selecting an entry in the list, both text fields will be filled.
Note that by pushing the **'CCU Nodes'** button, all selections other than the element configuration, mission name and system tree version selections, will be lost.

7.2.5.6 The 'CCU Versions' Button

The button is available when a CCU node has been selected, i.e. displayed in the **'CCU configuration name'** and the **'CCU pathname'** text items for current selections. When the button is pushed, the CCU versions

Copyright per DIN 34

based on the selected element configuration, mission name, system tree version and CCU node, are displayed in the list. By selecting an item in the list, it is displayed in the **'CCU Version'** text items for current selections. Double clicking in the list of CCU versions will have the same effect as a single selection.

7.2.5.7 The Selection List

There is a button for each level of the hierarchical organized CCU identifier. When any one of these is activated, the mission database will be requested to provide the list of identifier components corresponding to the label of the button (naturally using the already selected identifier components as input), and the received list will be displayed in the scrolling list item. Double clicking in the displayed list will be the same as selecting one item and then push the list button.

When making a selection in the component list, the button for the next lower level of the CCU identifier will be activated.

It is possible at any time to go back up in the hierarchy by activating one of the higher level buttons.

7.2.5.8 The 'Element Configuration' Text Item

The text item contains the current selection of the element configuration. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.9 The 'Mission Name' Text Item

The text item contains the current selection of the mission name. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.10 The 'System Tree Version' Text Item

The text item contains the current selection of the system tree version. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.11 The 'CCU Configuration Name' Text Item

The text item contains the current selection of the CCU configuration name. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.12 The 'CCU Pathname' Text Item

The text item contains the current selection of the CCU pathname. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.13 The 'CCU Version' Text Item

The text item contains the current selection of the CCU version. It is dimmed and cleared to indicate that a selection has not been performed.

7.2.5.14 The 'Apply' Button

When the window was invoked by the 'Multi Purpose Test Session' window .
When a CCU is identified (all the 'Current Selection' text items have been filled in), the **'Apply'** button will

Copyright per DIN 34

become active. By pushing **'Apply'**, the selected CCU is made available for the **'Multi Purpose Test Session'** window and its identification thereby displayed in the CCU text item of the window.
The **Select CCU** window is closed.

When the window was invoked by the 'Load Test Configuration' window.
When a CCU is identified (all the 'Current Selection' text items have been filled in), the **'Apply'** button will become active. By pushing **'Apply'**, the selected CCU is made available for the **'Load Test Configuration'** window and its identification thereby displayed in the CCU text item of the window.
The **Select CCU** window is closed.

7.2.5.15 The 'Reset' Button

The **'Reset'** button is active when changes have been made in any of the selections. By pushing the button, the CCU identifier initially displayed in the window will be displayed again.
The **'Reset'** button will be dimmed when the select CCU window is opened and after having activated reset. This will then indicate that the window content reflects the initially displayed state of the window.

7.2.6 The Create Session Window

7.2.6.1 General

The **Create Session** window will appear when the main window menu option **'Test Sessions–>Create'** is activated.

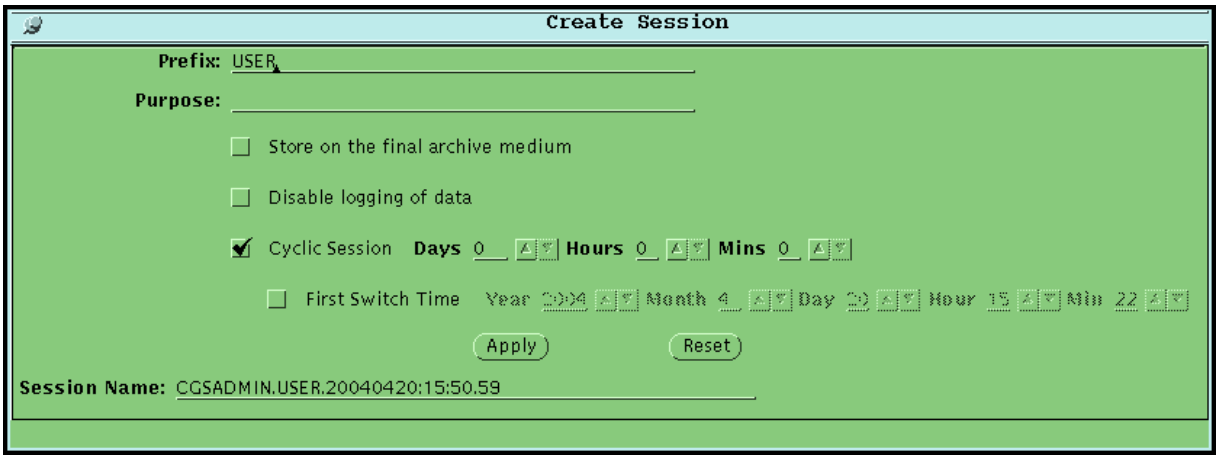

Create Session
Prefix: USER
Purpose:
Store on the final archive medium
Disable logging of data
Cyclic Session Days 0 Hours 0 Mins 0
First Switch Time Year 2004 Month 4 Day 20 Hour 15 Min 22
Apply
Reset
Session Name: CGSADMIN.USER.20040420:15:50.59

Figure 7–12 *: The Create Test Session Window*

All testdata generated during a test will go into the same 'envelope' represented by a test session created by the user for that particular test. The **'Create Session'** window is used to create sessions.

It is not mandatory to open a specific session, though. The database will accept test data and event log messages from the participating nodes. The data will go into a 'default test execution session' if there is no other session open.

At any time, there can be up to 5 tests going on in parallel, each with its own session open in the database, either the default session or a named session. Note that the default test session can only be used by one of the test executions.

7.2.6.2 The 'Prefix' and 'Purpose' Text Item

tbd

Copyright per DIN 34

7.2.6.3 The 'Prefix' Text Item

A prefix text string is currently configured as part of a session name. The prefix text string can be entered in this text item field.

7.2.6.4 The 'Purpose' Text Item

This text field corresponds to an attribute which the database will associate with the session. It is simply a descriptive text in free format.

7.2.7 The 'Store on the Final Archive Medium' Check Box

This check box can be used to start automatic final archiving (i.e. export).

7.2.7.1 The 'Disable Logging Of Data' Check Box

This check box can be used to disable logging. If checked, no test results will be saved.

7.2.7.2 The 'Cyclic Session' Check Box

This check box can be used to define a cyclic test session.

7.2.7.3 The 'First Switch Time' Check Box

tbd

7.2.7.4 The 'Apply' Button

When activating the **'Apply'** button, a test session is created according to the specified session name and final archiving selection. After creation of the test session, the window is closed.
The button is dimmed when the **'Session Name'** text field is empty.

The actions initiated when pressing the **'Apply'** button depend on whether the associated test configuration is in the state **Idle** or **Active**

When **Idle**, TSCV will interrogate DBS whether there already exists a session with that name. If not, the session name and the attributes will be stored in TSCV's internal database, and the session name will be displayed in the test configuration list in the main window. Otherwise, the user will be requested to change the session name. Later, when the user activates the Set–up function, TSCV will request DBS to create the session in the test result database.

When **Active,** TSCV will immediately request DBS to create the session in the test result database. Provided that this is successful, the session name will also be displayed in the test configuration list in the main window.

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

7.2.7.5 The 'Reset' Button

The button resets all changes entered in text fields and the' final archive medium' check box. The content of the **'Create Session'** window will be as when opened.

Copyright per DIN 34

7.2.8 The Multi Purpose Test Sessions Window

7.2.8.1 General

The **Multi Purpose Test Sessions** window, may be opened in three different contexts where the window is named:
– The *'Maintain Test Session'* window
– The *'Load Test Session to be Replayed'* window
– The *'Assign Replay Session to be Replayed'* window

Copyright per DIN 34

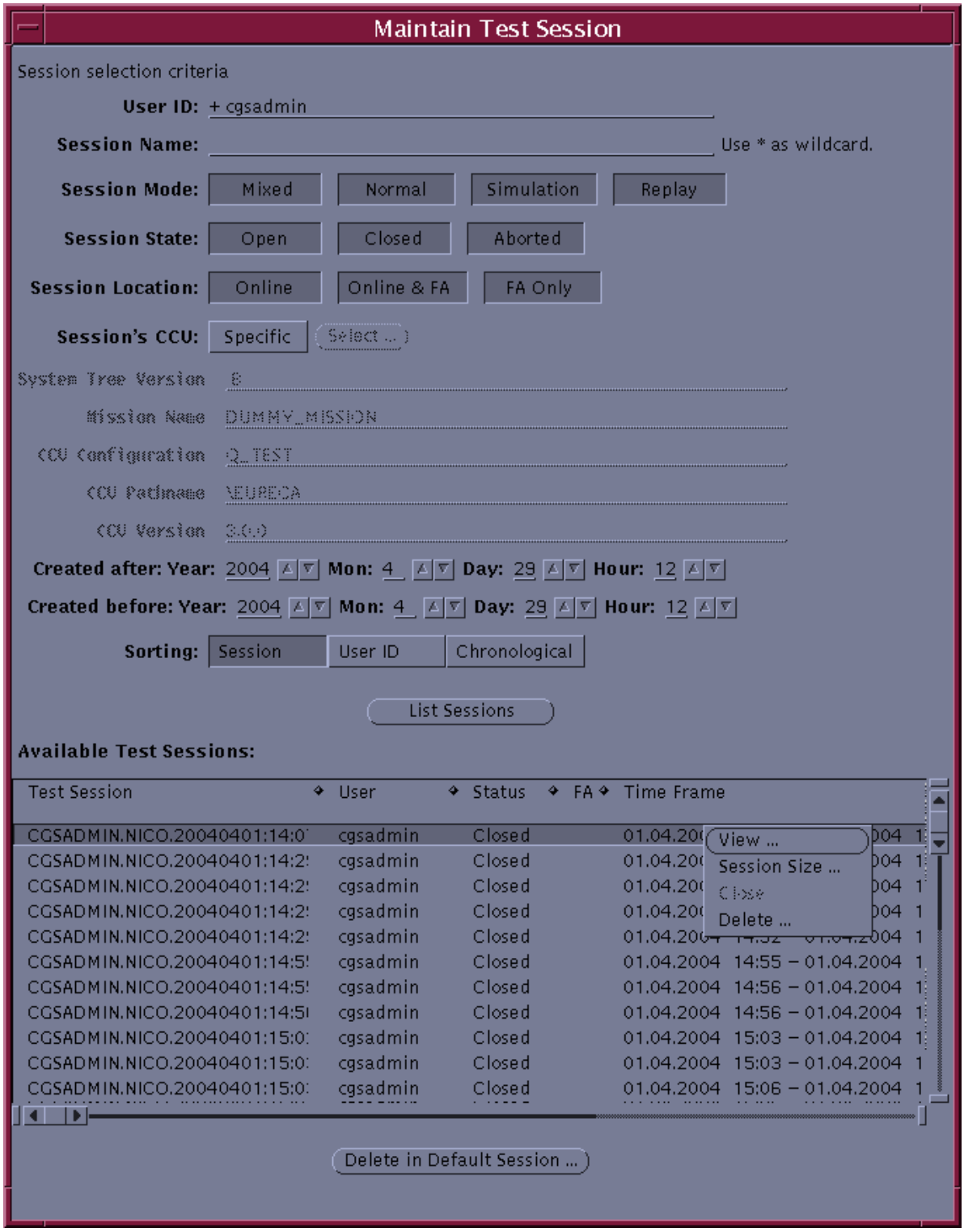

Maintain Test Session
Session selection criteria
User ID: + cgsadmin
Session Name:
Use * as wildcard.
Session Mode: Mixed Normal Simulation Replay
Session State: Open Closed Aborted
Session Location: Online Online & FA FA Only
Session's CCU: Specific Select ...
System Tree Version
Mission Name DUMMY_MISSION
CCU Configuration Q_TEST
CCU Pathname \EUPECA
CCU Version 3.0.0
Created after: Year: 2004 Mon: 4 Day: 29 Hour: 12
Created before: Year: 2004 Mon: 4 Day: 29 Hour: 12
Sorting: Session User ID Chronological
List Sessions
Available Test Sessions:
Test Session User Status FA Time Frame
CGSADMIN.NICO.20040401:14:0 cgsadmin Closed
CGSADMIN.NICO.20040401:14:2 cgsadmin Closed
CGSADMIN.NICO.20040401:14:2 cgsadmin Closed
CGSADMIN.NICO.20040401:14:2 cgsadmin Closed
CGSADMIN.NICO.20040401:14:2 cgsadmin Closed
CGSADMIN.NICO.20040401:14:5 cgsadmin Closed 01.04.2004 14:55 – 01.04.2004
CGSADMIN.NICO.20040401:14:5 cgsadmin Closed 01.04.2004 14:56 – 01.04.2004
CGSADMIN.NICO.20040401:14:5 cgsadmin Closed 01.04.2004 14:56 – 01.04.2004
CGSADMIN.NICO.20040401:15:0 cgsadmin Closed 01.04.2004 15:03 – 01.04.2004
CGSADMIN.NICO.20040401:15:0 cgsadmin Closed 01.04.2004 15:03 – 01.04.2004
CGSADMIN.NICO.20040401:15:0 cgsadmin Closed 01.04.2004 15:06 – 01.04.2004
View ...
Session Size ...
Close
Delete ...
Delete in Default Session ...

Figure 7–13 : *The Maintain Test Session Window*

When the window is opened, the filter criteria will be as when last opened. One exception is the text fields identifying the CCU, which will contain the CCU of the test configuration which is in focus in the main window. The different parts of the window are:

The Session List Filters

The session list filters are used to restrict which sessions to be retrieved from the test result database. The

Copyright per DIN 34

filter includes specification of the following parameters:
– session pattern
– session mode
– session status
– CCU
– time span (*'created after'* to *'created before'*)
– sorting criteria

List Sessions
When operating the **'List'** button, the subset of sessions fulfilling the filter parameters, will be displayed in the scrolling list.

Operations on a Selected Session
Two buttons are available to operate on the session currently selected in the list of sessions. When pushing the **'View...'** button, the key data of the selected session will be viewed in a text window.

The other button, also named the <action> button, will have different labels and and initiate different functions dependent on the context the window is opened in:

The *'Maintain Test Session'* window:
When the **'Delete'** button is activated, the currently selected test session is deleted after user confirmation.

The *'Load Replay Session'* window:
When the **'Load'** button is activated, the embedded data of the currently selected test session is loaded into the currently selected slot in the main window preparing for replay of the recorded data of the test session.

The *'Assign Replay Session'* window:
When the **'Assign'** button is activated, the embedded data of the currently selected test session is assigned to the test configuration currently selected slot in the main window preparing for replay of the recorded data of the test session.

A third button will be available in the *'Maintain Test Session'* window. This button is named **'Close'**, and is used to close sessions open in the Test Result Database, but not known to TSCV. The button will be dimmed when the selected session is listed in the test configuration list or when the session state is different from *'Open'*.

The Default Session
The window provides one operation on the default session: To delete data in the default session with a time stamp older that the one specified by the **'Created before'** text fields.

7.2.8.2 The 'Session Pattern' Text Item

This part of the session list filter is used to restrict the names of the sessions to be retrieved. The text field is used to specify a session name pattern that has to match for the test sessions to be listed. By specifying the wild–card , '%', there will be no restrictions on the session names.

7.2.8.3 The 'Session Mode' Choice

This part of the session filter is used to restrict the mode of the sessions retrieved to be displayed in the **'Selected Sessions'** list. The choice is non–exclusive.

Copyright per DIN 34

Mixed
Sessions containing nodes with different modes will be displayed.

Normal
Sessions with session mode **'Normal'** and **'Mixed'** will be displayed. The session mode **'Normal'** implies that the mode of all the TES instances, participating in the test execution, is **'Normal'.**

Simulation
Sessions with session mode **'Simulation'** and **'Mixed'** will be displayed. The session mode **'Simulation'** implies that the mode of all the TES instances, participating in the test execution, is **'Simulation'**.

The *'Delete Test Session'* window
Replay
Sessions with session mode **'Replay'** and **'Mixed'** will be displayed. The session mode **'Replay'** implies that the mode of all the TES instances, participating in the test execution, is **'Replay'.**

7.2.8.4 The 'Session Status' Choice

The *'Delete Test Session'* window
The choice is non–exclusive and determines the status of the test sessions to be listed.

Open
By selecting the **'Open'** choice, the named sessions which are currently open, will be displayed. TSCV is able to manage up to 5 simultaneously open named sessions (in addition, there will be a default session, which is not displayed in the list).

Closed
By selecting the **'Closed'** choice, the sessions which are currently closed, will be displayed.

The *'Load Replay Session'* window and the *'Assign Replay Session'* window
The choice is always dimmed. It is set to **'Closed'** by default, because data can only be replayed for closed test sessions.

7.2.8.5 The 'Specific' Choice

When a session is created in the *Test Result DataBase*, it will be associated with a CCU via the test configuration representing the current test system set–up. It is possible to restrict the listed sessions to those created for a specific CCU. When the **'Specific'** choice is activated, the dimming of the belonging text fields and button will cease. The database will then return only the sessions associated with the CCU identified in the text fields.

7.2.8.6 The 'Select...' Button

The **'Select...'** button will become active when the session list filter **'Session's CCU'** is set to **'Specific'**. By operating the button, the **'Select CCU'** window is opened and that window may be used to select a CCU to be used as a session list filter.

Copyright per DIN 34

7.2.8.7 The 'System Tree Version' Text Field

The text field displays the **'System Tree Version'** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **'List'** button.

7.2.8.8 The 'CCU Configuration' Text Field

The text field displays the **'CCU Configuration'** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **'List'** button.

7.2.8.9 The 'CCU Pathname' Text Field

The text field displays the **'CCU Pathname'** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **'List'** button.

7.2.8.10 The 'CCU Version' Text Field

The text field displays the **'CCU Version'** part of the currently selected CCU identifier, which may be used to restrict the content of the list. When the text item is dimmed, it indicates that there will be no CCU restrictions as part of the session list filter when operating the **'List'** button.

7.2.8.11 The 'Created After' Timestamp

When the window is opened, the **'Created after'** timestamp is set to beginning of 1995, i.e. no time restriction. It is possible to restrict the time span with the individual numeric text fields for year, month, day and hour. Only sessions created after the indicated point in time, will be displayed in the list.

7.2.8.12 The 'Created Before' Timestamp

When the window is opened, the **'Created before'** timestamp is set to the moment when the window was opened i.e. no time restriction. It is possible to restrict the time span with the individual numeric text fields for year, month, day and hour. Only sessions created before the indicated point in time, will be displayed in the list.

Please note that the point in time specified for **'Created before'** will also apply when deleting data from the default test execution session.

7.2.8.13 The 'Sorting' Choice Options

The options are used to determine sorting of the test sessions to be listed in the panel '**Available Test Sessions:**' when pushing '**List Sessions**'. There are only three options, **'Session', 'User ID'** and **'Chronological'.**

7.2.8.14 The 'List Session' Button

When the **'List Session'** button is pressed, the selection filter will be submitted to the database, which will return a list of sessions. The content of the scrolling session list will be replaced by the acquired session list in the panel below.

Copyright per DIN 34

7.2.8.15 The ‘Test Sessions’ List Panel

The scrolling list in this window will display sessions retrieved from the test result database when the button labelled **’List Session’** is pressed. Usually, it is required to select a restricted set of sessions.
TSCV will provide a selection filter to the database, and the selection will be restricted by the values specified for the filter items. A pop–up menu with four menu buttoms can be invoked when selecting a test session from the displayed list.

7.2.8.16 The ’View...’ Button

For each test execution session, the database keeps a key record (’session info record’). By pressing the ’**View...’** button, the session key data of the session currently selected in the **’Select sessions’** list , will be displayed in a separate text window.

The button will be dimmed when the **’Select sessions’** list is empty.

7.2.8.17 The ’Session Size ...’ Button

tbd

7.2.8.18 The ’Close’ Button

This button is used to close sessions open in the Test Result Database, but not known to TSCV. The button will be dimmed when the selected session is listed in the test configuration list or when the session state is different from *’Open’*.

7.2.8.19 The ’Delete ...’ Button

When the **‘Delete’** button is pressed, it is requested to delete the session currently selected in the **’Select Sessions’** list. Before deleting the session, the user is asked to confirm the delete request. When the request is acknowledged, the complete ’session envelope’ (the session key record and all test data acquired into it) identified by the selected session name, is deleted.

The button will be dimmed when the **’Select Sessions’** list is empty.

7.2.8.20 The ’Delete in Default Session...’ Button

When the **‘Delete in Default Session...’** button is pressed, a notice window occurs (see Figure 7–14).

TSCV will take the time specified by the **’Created before**’ items, format it and display it in a confirm window. The user will be requested to confirm the wish to delete old data acquired into the default test execution session – data acquired before that point in time. The request will then be submitted to the database for execution.

Copyright per DIN 34

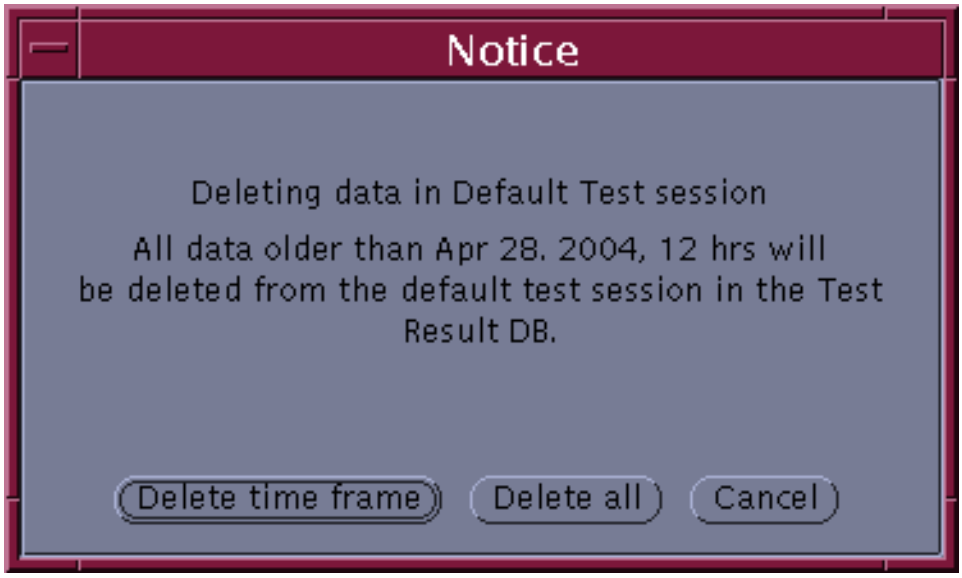

Notice
Deleting data in Default Test session
All data older than Apr 28, 2004, 12 hrs will be deleted from the default test session in the Test Result DB.
Delete time frame
Delete all
Cancel

Figure 7–14 *: Delete Data in Default Test Session Notice Window*

If one does not use always named sessions, but (as an implicit result of this) the default test session instead, it is possible that the reserved table space for the default test session becomes completely occupied. It is thus necessary to delete the entries in the default test session from time to time. To achieve this, use the **Sessions–>Maintain...–>Delete in Default Session...** *function on a regular basis. Otherwise, DBS will report database (Oracle) errors when the event table becomes full.*

7.2.9 The Node Property Sheet Window

7.2.9.1 General

The **'Node Property Sheet'** window will be opened when the the **'Properties–>Node...'** menu choice is activated or when double clicking in the node list in the main window. Note that when more than one node is selected in the list of nodes, properties of the first selected node will be displayed when selecting the **'Properties–>Node...'** menu choice.
The **'Node Property Sheet'** contains entries for a set of attributes related to one single selected node, and which will be acted upon by TSCV when the user commands TSCV to execute the set–up function.

Copyright per DIN 34

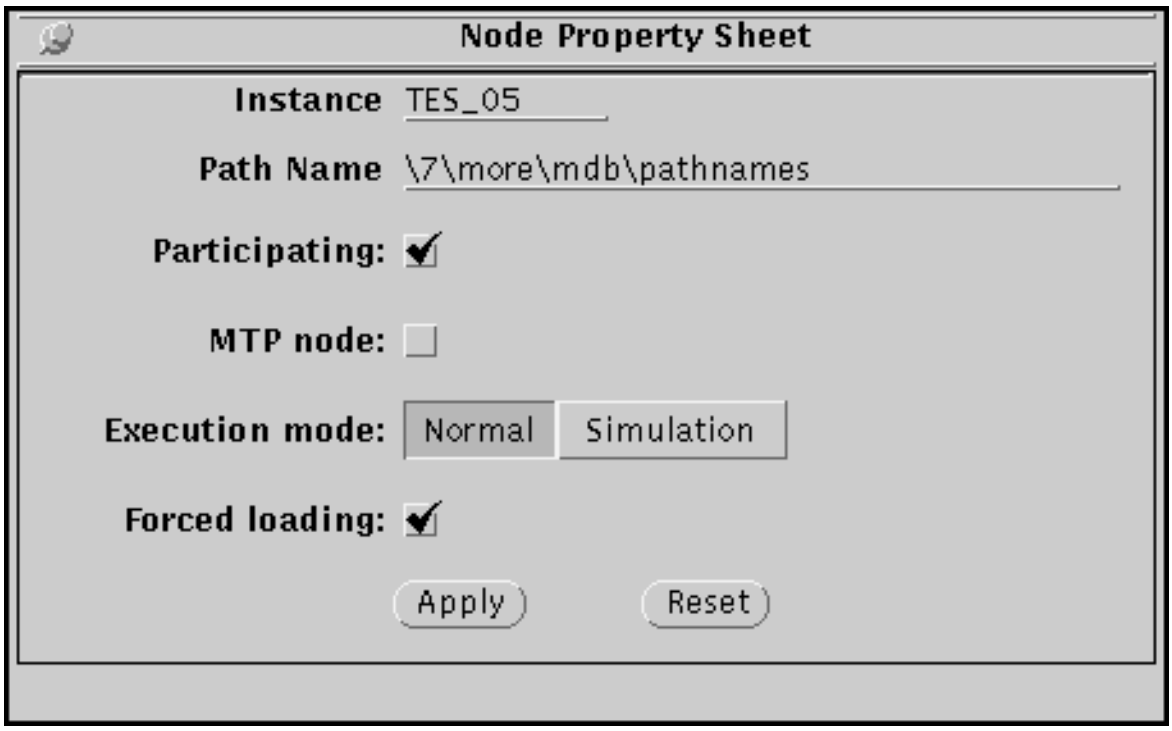


Figure 7–15 *: The Node Property Sheet Window*

In the **'Node Property Sheet'** the user can:

- Edit the node attributes for the nodes of an **'Idle'** test configuration (one node entry at a time).
- View the attributes for the participating nodes of an **'Active'** test configuration (one node entry at a time)

Hence, the **'Node Property Sheet'** will open in read only modus (with dimmed window components) when the test configuration in focus is **'Active'**, and in edit modus when the status **'Idle'**.

Both test nodes (TES), workstation nodes (HCI) and simulation nodes (CSS) can be viewed. Only the 'Participating flag' can be edited for HCI nodes, i.e. the rest of the attributes will be dimmed. The CSS node can only be viewed, i.e. no attributes can be changed. For TES nodes, all attributes used in the current CGS version, will be available.

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

7.2.9.2 The 'Instance' Text Field

The text field contains the physical name of the node. The attributes of that node is viewed in the **'Node Property Sheet'**. The text field is not editable, i.e. attributes for another node can not be viewed by entering a new name into the text field.

7.2.9.2.1 The 'Path Name' Text Field

The text field contains the full path name of the node. The text field is not editable.

Copyright per DIN 34

7.2.9.3 The 'Participating' Check Box

The **'Participating'** checkbox is used to determine whether or not the selected node shall participate in the test when the **'Idle'** test configuration is activated (setup is performed). The **'Participating'** flag can be set for test nodes and workstation nodes. It is dimmed when the node is participating in an **'Active'** test configuration and when the selected node is the MTP node (which can not be made non–participating).

7.2.9.4 The 'MTP node' Check Box

One of the participating nodes of a test configuration must play the role of the Master Test Processor (MTP). Only participating test nodes can be selected as the single MTP and only one of the participating test nodes can be the MTP. When selecting a new node as the MTP node of the test configuration, the previously selected node will automatically be deselected. No actions from the user is required to ensure that only one test node is selected as the MTP node.
When trying to set a non–participating test node as the MTP by pressing the **Apply** button, it will be denied without any notice to the user.

The **MTP** check box is applicable for test nodes only, and will be dimmed for other types of nodes, for the MTP node and when the node is participating in an **Active** test configuration (then the MTP status cannot be changed).

7.2.9.5 The 'Execution Mode' Check Box – Normal or Simulation

The **Execution Mode** check box is used to define the execution mode of the test node. The test node can execute in the modes **Normal** or **Simulation**.

The **'Execution Mode'** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing properties is not allowed)

7.2.9.6 The 'Execution Mode' Check Box – Replay TCs

The **'Execution Mode'** check box is used to define the execution mode of the test node. The test node can execute in the modes **Replay** or **Simulation**. If there is not connected any replay data to the node, the only valid mode is **Simulation**. Then the check–box is dimmed.

The **'Execution Mode'** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing properties is not allowed)

7.2.9.7 The 'Forced Loading' Check Box

The **Forced loading** check box is used to determine whether or not loading of the database is required when the test node is initiated (set up). This is required whenever the scope of the data to be loaded has changed (CCU or data load point).

The **Forced loading** check box is applicable for test nodes only, and will be dimmed for other types of nodes and when the test node is participating in an **Active** test configuration (then changing the property is not legal)

7.2.9.8 The 'Apply' Button

When activating the **'Apply'** button, the content of the **'Node Property Sheet'** is applied as properties for the selected node.

Copyright per DIN 34

Please note that any modifications made in the window will be completely local to the window until the **Apply** button is pressed.

7.2.9.9 The 'Reset' Button

The button resets all changes. The content of the **'Node Property Sheet'** window will be as when opened. The viewed attributes will then display the values currently applied on the selected node.

7.2.10 The Replay Properties Window

The **'Replay Properties'** window will be opened when the menu option **'Replay–>Properties'** is activated. It contains entries for a set of attributes related to a replay set–up, and which will be acted upon by TSCV when you activates the set–up function.

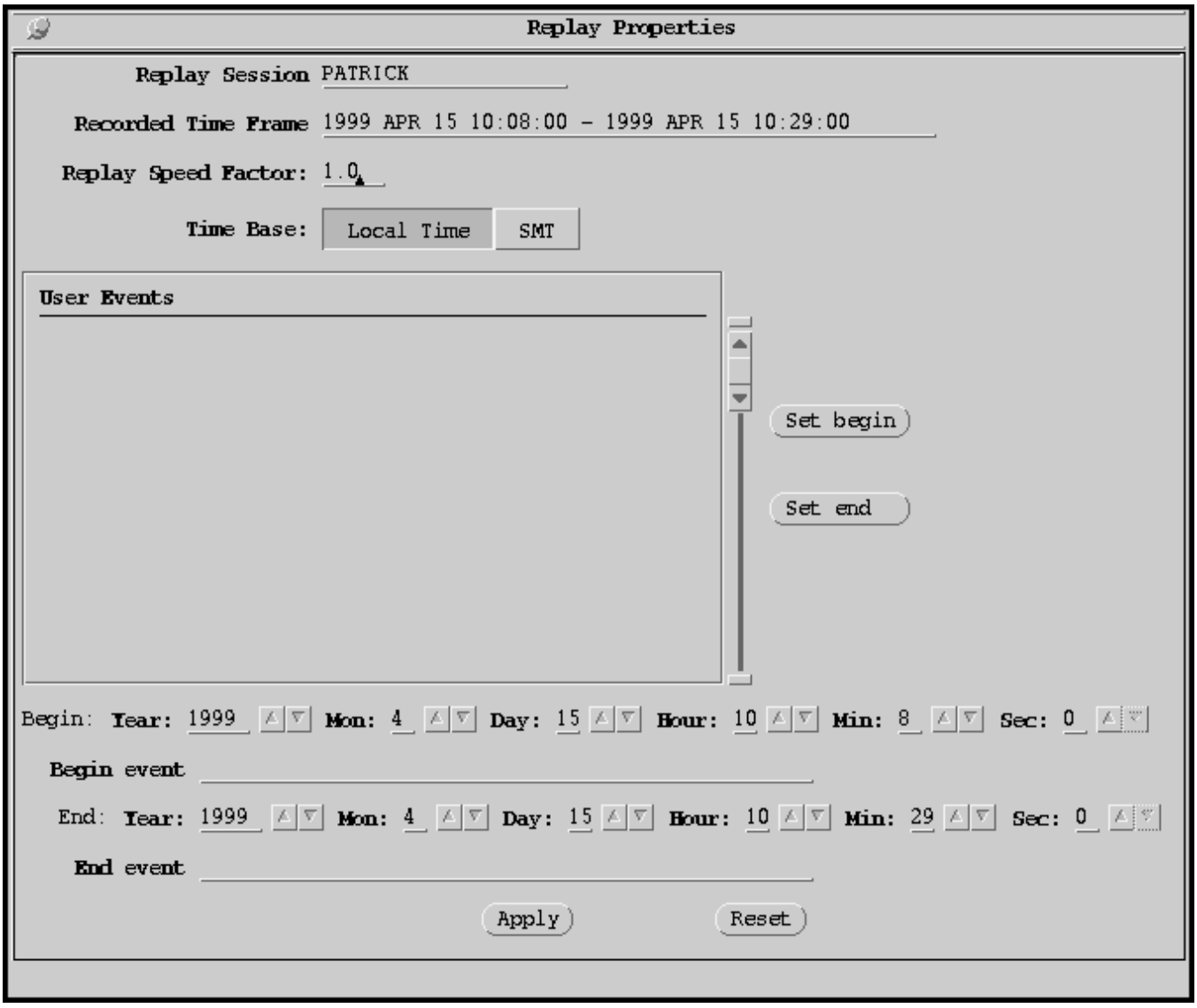

Replay Properties
Replay Session PATRICK
Recorded Time Frame 1999 APR 15 10:08:00 - 1999 APR 15 10:29:00
Replay Speed Factor: 1.0
Time Base: Local Time SMT
User Events
Set begin
Set end
Begin: Year: 1999 Mon: 4 Day: 15 Hour: 10 Min: 8 Sec: 0
Begin event
End: Year: 1999 Mon: 4 Day: 15 Hour: 10 Min: 29 Sec: 0
End event
Apply
Reset

Figure 7–16 *: The Replay Properties Window*

In the *'Replay Properties'* window you can:

Copyright per DIN 34

– See which session has been attached to the test configuration slot for replay purpose.
– See the total available time span (in local time) for the session to be replayed.
– See the user events from the session to be replayed.
– Set replay speed factor.
– Set replay begin and end times by selecting user events.
– Set replay begin and end times in numeric fields representing timestamps.
– Switch between Local Time and Simulated Mission Time representation.

The *'Replay Properties'* window can only be opened when a replay session has been attached to the selected test configuration. It will be opened in read only mode (with dimmed window components) when the test configuration in focus is **'Active'.**

When the *'Replay Properties'* window is open, its contents will be updated when you set another test configuration in focus. Any changes you made, which you have not confirmed by pressing the **'Apply'** button, will then be lost.

7.2.10.1 The 'Replay Session' Text Field

The textfield contains the name of the recorded session which you have selected and attached to the test configuration in focus as a session to be replayed. The text field is not editable.

7.2.10.2 The 'Recorded Time Frame' Text Field

The textfield contains the available time frame of the recorded session which you have selected and attached to the test configuration in focus as a session to be replayed, i.e. the session creation time and the session close time. The text field is not editable.

7.2.10.3 The 'Replay Speed Factor' Text Field

The textfield contains the speed factor which the *Test Execution Software* shall use while replaying data from the session which you have selected and attached to the test configuration in focus as a session to be replayed.

You may specify the value in the range from 0.1 to 100.0. A range and format check will be performed when you hits the **'Enter'** key on your keyboard, and when you activate the **'Apply'** button.

7.2.10.4 The 'Time Base' Choice

You should use this choice item to switch the time base for the replay time frame.

The specified value (**Local Time** or **Simulated Mission Time**) will be used by TSCV when setting up the test configuration. It will be used as part of a search key when retrieving replay data (archive files) to be provided to the *Test Execution Software*.

The setting will also be reflected in the **'Replay Properties Sheet'** itself: When you select a user event either as a begin time or an end time, the associated time–stamp will be filled in into the corresponding numeric fields. Depending on the current time base setting, either the associated **Local Time** or the **Simulated Mission Time** will be shown.

If the current **Begin Time** or/and **End Time** is/are set via a selected user event (without subsequent editing

Copyright per DIN 34

of the numeric fields), the displayed time–stamp(s) will change and display the event's time in the correct time base.

7.2.10.5 The 'User Events' List

The list contains the user events belonging to the test session which you have selected and attached to the test configuration in focus as session to be replayed. The events' *'Short Text'* and hour:minute part of the timestamp is viewed in the list (the clock reading be taken from **Local Time** or **Simulated Mission Time** – according to the current setting).

When you activate either the button **'Set begin'** or **'Set end'**, the point in time when the event was created will be taken as replay begin or end time, correspondingly. The event's *'Long Text'* will be entered into the corresponding text field, and the event's timestamp will be entered into the corresponding numeric fields. Depending on the time base setting, the time shown will be either the Local Time or the Simulated Mission Time reading when the event was created.

7.2.10.6 The 'Set Begin' Button

You should use this button to specify a user event representing the replay begin time, i.e. the oldest recorded and archived data which will be replayed by the *Test Execution Software*.

When you activate the button, the currently selected event will be taken to represent the begin time. The event's *'Long Text'* will be entered into the **'Begin event'** text field, and the event's time–stamp will be entered into the 'Begin time' numeric fields. Depending on the time base setting, the time shown will be either the **Local Time** or the **Simulated Mission Time** reading when the event was created.

7.2.10.7 The 'Set End' Button

You should use this button to specify a user event representing the replay end time, i.e. the latest recorded and archived data which will be replayed by the Test Execution Software.

When you activate the button, the currently selected event will be taken to represent the end time. The event's **'Long Text'** will be entered into the **'End event'** text field, and the event's time–stamp will be entered into the **'End time'** numeric fields. Depending on the time base setting, the time shown will be either the **Local Time** or the **Simulated Mission Time** reading when the event was created.

7.2.10.8 The 'Begin' Timestamp

The numeric fields contain the current **'Begin Time'**, i.e. the time–stamp from when the *Test Execution Software* will replay the archived data.

When you edit the **'Begin Time'** by modifying one of the numeric fields, the possible association to a selected user event is lost (reflected by the clearing of the long event text).

Each time you change value in one of the **'Begin Time'** numeric fields, a validity check is made. If the time would become greater than the current **'End Time'**, the change will be ignored. If the timebase is **'Local Time'**, and the time would be less than the recorded session's creation time, the change would be ignored.

7.2.10.9 The Begin 'Event' Text Field

This text field will contain the long text of the user event you have selected to reptresent the replay **'Begin Time'**. If you set or modify the **'Begin Time'** by changing value in one of the **'Begin Time'** numeric fields, the association to the selected user event is lost, and consequently the text field is cleared.

Copyright per DIN 34

7.2.10.10 The 'End' Timestamp

The numeric fields contain the current **'End Time'**, i.e. the time–stamp up till when the Test Execution Software will replay the archived data.

When you edit the **'End Time'** by modifying one of the numeric fields, the possible association to a selected user event is lost (reflected by the clearing of the long event text).

Each time you change value in one of the **'End Time'** numeric fields, a validity check is made. If the time would become less than the current **'Begin Time'**, the change will be ignored. If the timebase is **'Local Time'**, and the time would be greater than the recorded session's close time, the change will be ignored.

7.2.10.11 The End 'Event' Text Field

This text field will contain the long text of the user event currently selected to reptresent the replay **'End Time'**. If you set or modify the **'End Time'** by changing value in one of the **'End Time'** numeric fields, the association to the selected user event is lost, and consequently the text field is cleared.

7.2.10.12 The 'Apply' Button

When you activate this button, settings you have made in the window will be saved.

7.2.10.13 The 'Reset' Button

You may abandon changes you have made in the window by activating this button. The values will be reset to the settings valid last time you activated the **'Apply'** button (or when the recorded session was attached to the test configuration slot).

7.2.11 The Maintain System Topology Window

7.2.11.1 General

When the *'Maintain System Topology'* window is opened and after **'Reset',** the *'System Topology Table'* residing at *'$CGS_HOME/config'* will be loaded and all entries put into the list of *'System Topology Table'* entries.

The user may then update the content of the list by first entering values into the text fields **'Host'**, **'Instance'** and **'Port Number'** (only for SAS) and then request to add an entry or change the currently selected entry. The currently selected entry can also be deleted.

Please note that any modifications made in the window will be completely local to the window until the **'Apply'** button is pressed. Also note, that when selecting an entry in the list, the content of the newly selected item will be displayed in the text fields **'Host'**, **'Instance'** and **'Port Number',** and any changes made in the text items will be overwritten and lost.

When the user wants to update the *'System Topology Table'* of the test system, the **'Apply'** button must be pushed. The consistency of the content of the list of *'System Topology Table'* entries, is checked. When errors are found, this is reported by error messages and applying changes will be denied. When the consistency check is successful, the user can select either to **'Store and Shutdown'**, **'Store'** or **'Cancel'**.

The applications, including TSCV, loads the *System Topology Table* when invoked. Therefore, a test system shutdown has to be performed to ensure that changes will be read by the applications throughout the test system.

Copyright per DIN 34

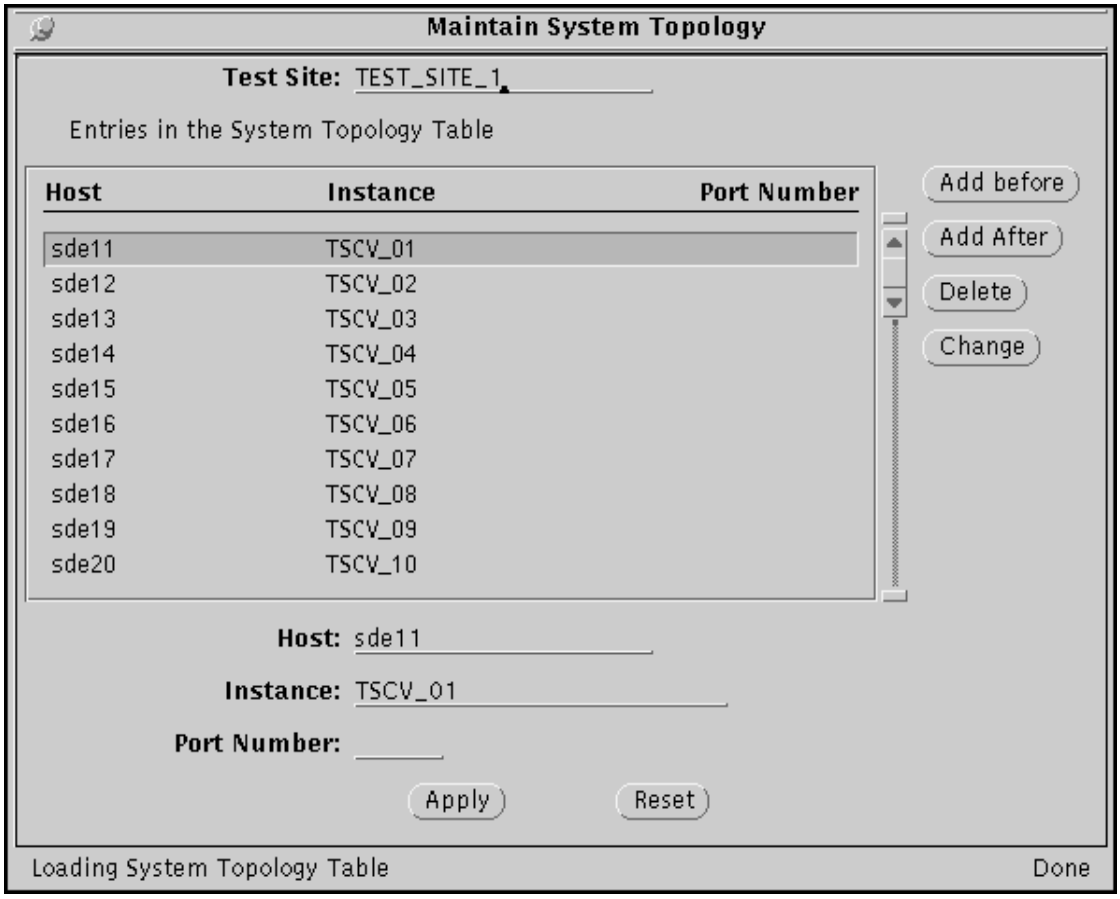


Figure 7–17 *: The Maintain System Topology Window*

7.2.11.2 The 'Test Site' Text Field

The **'Test Site'** text field is an editable text item for display and edit of the test site. Changes are only local until a new *'System Topology Table'* is stored after applying changes.

7.2.11.3 The System Topology Table List

The **'System Topology Table'** list will be a list of all entries in the *'System Topology Table'* when the *'Maintain System Topology'* window is opened and after **'Reset'**. The user may update the content of the list by entering values into the text fields **Host**, **Instance** and **Port Number** (only for SAS) and then request add or change. System Topology Table entries are listed in the same sequence as found in the *'System Topology Table'*, i.e. no sorting will be provided. The columns are:

- Host name
- Instance (type and instance)
- Port number (only applicable for the process type SAS).

No operation is attached to double clicking an entry in the list
Changes are only local until a new *'System Topology Table'* is stored after applying changes, and then the consistency will be checked.

7.2.11.4 The 'Add Before' Button

The **'Add Before'** button is used to add a new entry into the displayed list of *'System Topology Table'* entries before the currently selected one. The content of the **'Host'**, **'Instance'** and **'Port Number'** text fields are

Copyright per DIN 34

validated according to their validation rules. If the content is not accepted, the user is notified and adding a new entry is denied.

7.2.11.5 The 'Add After' Button

The **'Add After'** button is used to add a new entry into the displayed list of *'System Topology Table'* entries after the currently selected one. The content of the **'Host'**, **'Instance'** and **'Port Number'** text fields are validated according to their validation rules. If the content is not accepted, the user is notified and adding a new entry is denied.

7.2.11.6 The 'Delete' Button

The **'Delete'** button is used to delete the currently selected entry in the list of *'System Topology Table'* entries. The entry will be removed from the list without user confirmation.

7.2.11.7 The 'Change' Button

The **'Change'** button is used to change the currently selected entry in the list of *System Topology Table* entries. The content of the **Host**, **Instance** and **Port Number** text fields are validated according to their validation rules. If the content is not accepted, the user is notified and changing the currently selected entry is denied.

7.2.11.8 The 'Host' Text Field

The **'Host'** text field is used to display and edit the host name. When selecting an item in the list of *'System Topology Table'* entries, the host of the currently selected item is written into the text field.

The user is free to edit the text field in order to add or change entries in the list of *'System Topology Table'* entries. There will be no checking on the input at the time the list is updated. When selecting **'Apply'**, consistency checks will be performed.

7.2.11.9 The 'Instance' Text Field

The '**Instance**' text field is used to display and edit the process type and instance. When selecting an item in the list of *'System Topology Table'* entries, the instance of the currently selected item is written into the text field.

The user is free to edit the text field in order to add or change entries in the list of *'System Topology Table'* entries. There will be no checking on the input at the time the list is updated. When selecting **'Apply',** consistency checks will be performed.

TSCV will recognize the process type depending on their names. The following list shows how the different applications will be recognized:

HCI_XX – recognized as a HCI application.

TES_XX – recognized as a TES application.

TSCV_XX – recognized as a TSCV application.

TEV_XX – recognized as a TEV application.

DBS_XX – recognized as a DBS application.

Copyright per DIN 34

CIS_XX – recognized as a CIS application.

other – All other names is assumed to be SAS applications.

where XX must be a number between 01 and 32.

7.2.11.10 The 'Port Number' Text Field

Port numbers should not be used any longer in the System Topology Table!

7.2.11.11 The 'Apply' Button

The **'Apply'** button is used to update the System Topology Table used by the test system. When pushed, the content of the list of *'System Topology Table'* entries are interpreted and a local system topology table is generated. The consistency is checked (see below). If everything is accepted, a warning dialog gives the user three options:

- **Store and Shutdown**
  The test system has to be restarted before the new *'System Topology Table'* will be used. An application shutdown will be performed without any further user confirmation. Active test configurations will be stopped. At last, TSCV is stopped.
- **Store**
  A new *'System Topology Table'* will be stored. Applications must later be restarted before the changes will take effect.
- **Cancel**
  The apply request is cancelled.

The consistency checks performed upon **'Apply'**, are:

- The instances (process type and instance) shall be unique for process types TES, HCI, TSCV, TEV and CSS.
- The SAS names specified for one host, shall be unique.
- Up to 10 DBS instances, all on the same node.
- Only one of HCI, TES, TSCV, TEV, CSS can be hosted on a node, but they can share the same node and also be hosted on the same node as DBS.
- The number of instances is limited to 32 for TES, HCI, TSCV, TEV and CSS.
- The port numbers specified for one host, shall be unique. The limitation rule for port numbers implies that there will be a maximum of 20 SASes per node.

7.2.11.12 The 'Reset' Button

The **'Reset'** button resets all changes. The content of the *'System Topology Table'* is loaded and displayed. All changes done locally in the *'System Topology Table'* entries, will be lost.

Copyright per DIN 34

7.2.12 The Display Request Window

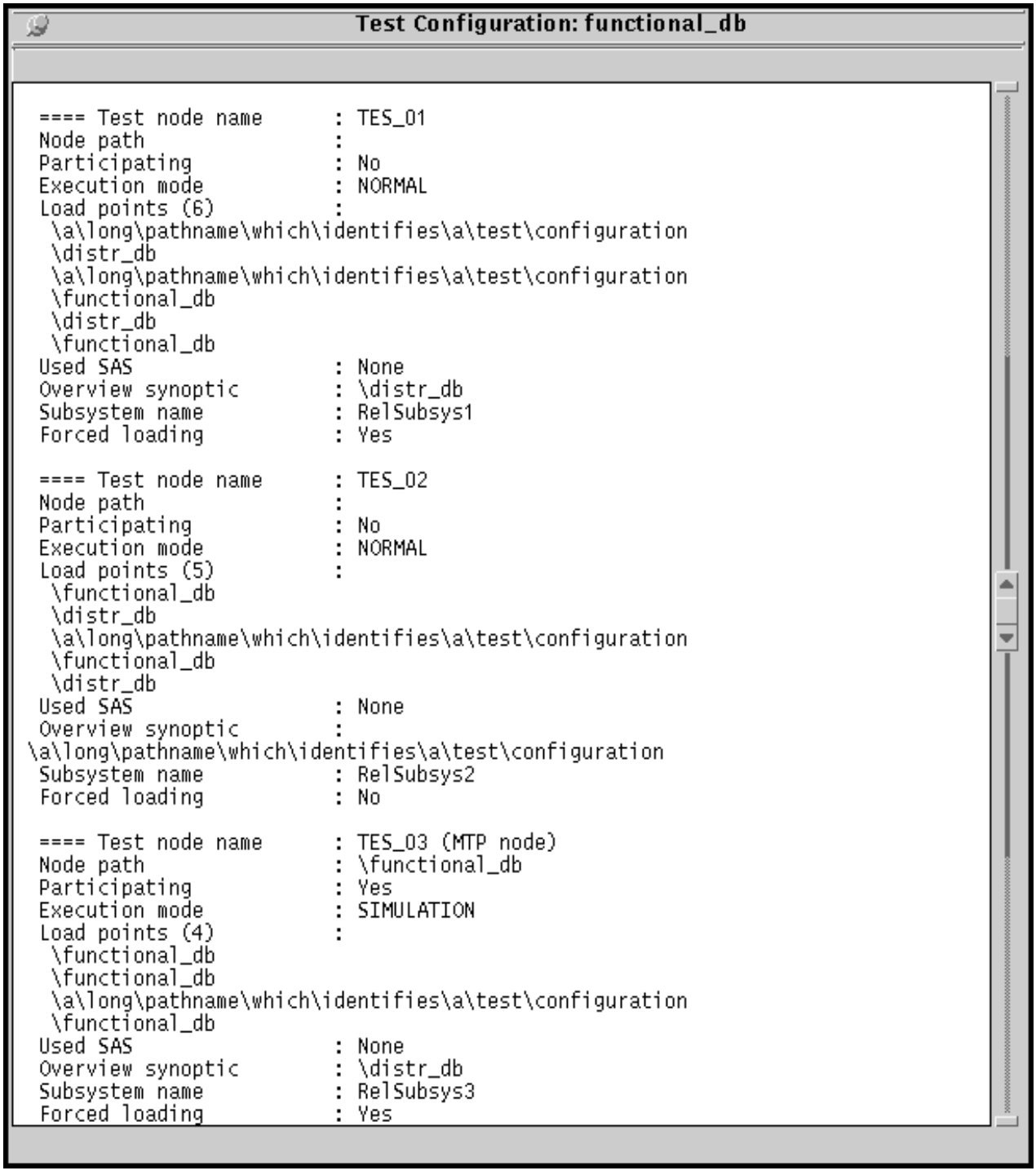


Figure 7–18 *: The Display Request Window*

Whenever the user has issued a **View** operation on a selected object, the content of the object will be viewed in the TSCV Display Request Window shown in figure 12. A view operation is either operating a **View** button or to double click in a list. These **View** operations exists:

- **View test configuration** currently selected in the TSCV main window
- **View test configuration** currently selected in the *'Load test configuration'* window.

Copyright per DIN 34

- **View session key data** in the ’Multi Purpose Test Sessions’ window.
- **View software versions** as defined in the version id table.

The window can be closed by pressing the window pin. Each time a ’view’ operation is activated, a new Display Request window will be opened. The window has a scrollbar, allowing the user to scroll through the buffer.

The user can not edit the content of the window neither save it on file. A selected part of the window content can be copied, however, and pasted into another window, e.g. as text editor.

Each view request will be displayed in a separate Display Request window. This behavior can be utilized to e.g. compare test configurations.

7.2.13 Operator Commands and Operations

7.2.14 TSCV Invocation, Interactive Mode

TSCV has to be invoked as an Unix process by a Unix command. In the integrated CGS, Test System Setup can be selected in a TLUI menu, and TLUI will give the necessary Unix command. It can also be started from the command line of an xterm or command tool window. TSCV will act upon a set of command line parameters.

When invoking TSCV in the interactive mode, it is possible to specify command line options. The command line options are optional.

The command line options that can be used for TSCV, are:

- –geometry {WxH}{[+|–]x[+|–]y}
- –WG {WxH}{[+|–]x[+|–]y} , (short notation for –geometry)
- –position x y , (note that position specified by –geometry has priority).
- –Wp x y, (short form for –position)
- –size w h , (note that size specified by –geometry has priority).
- –Ws w h, (short form for –size)
- –Wi, TSCV comes up open (default behaviour)
- +Wi, TSCV comes up closed (iconified)
- –WP x y , the position if the TSCV icon

An example of the syntax when including the **–geometry** command line option:

start_tscv –geometry +50+75

In fact, also the window size can be specified (although this will hardly be of any use); the syntax is then

start_tscv –geometry 500x400+50+75.

Copyright per DIN 34

If position is not provided on the command line, TSCV will search the X resource database. It will search for the resource variable **tscv.geometry**, and the variable string should be formatted as **wwwwxhhhhh+xxxx+yyyy**

7.2.15 TSCV Invocation, Batch Mode

Executing TSCV in batch mode has to be signalled via command line arguments. The following pairs of argument designators and argument values have to be provided, optional arguments in brackets:

- mn <mission_name>
- ec <element_configuration>
- sv <system_tree_version>
- sn <system_tree_node_name>
- cn <CCU_name>
- cv <CCU_version>
- tc <test_configuration>
- [–ts <test_session_name> [–fa]]
- [–q]

If one or more of the parameters are found to be invalid, TSCV will terminate, providing an exit code identifying the problem.

The set of ’batch parameters’ identifying the CCU and test configuration are mandatory:

<mission_name>	is the name of the mission, e.g. ”FLIGHT5711”
<element_configuration>	is the name of the element configuration, e.g. ”APM”
<system_tree_version>	is the identification of the version of the system name tree to be used for the given mission an element configuration, e.g. ”5”
<system_tree_node_name>	is the virtual node in the system (name) tree for the selected element configuration, mission and version of the name tree where the desired CCU is looked up, e.g. ”\APM\EGSE”
<CCU_name>	is the name of the CCU in MDB, e.g. ”IF_TEST”
<CCU_version>	is the version identification of this CCU. Note that a full version identification comprises the version, issue and revision number, e.g. ”1.1.0”
<test_configuration>	is the pathname of the database end item of type EGSE_TEST_CONFIGURATION. This end item describes how the setup of the test equipment shall be done, e.g. ”\APM\EGSE\SYSTEM\IF_TEST\SETUP_3”

Please note that the MDA **pathnames** on the Unix command line must be **enclosed by single quotes.**

The rest of the parameters are optional.

<test_session_name>	is the name of the test session to be created in the database for the given test,e.g. ”IF_TEST_3” (if test_session_name is omitted, all test result data will go into the default test session).

If the **<test_session_name>** parameter is omitted, data produced during the test will go into the ’universal’ default test execution session in the database. Note, however, that the default test execution session can only be used by one test configuration.

Copyright per DIN 34

The optional parameters **fa** and **q** are used as follows:

fa this argument does not have any meaning unless a test session name is provided. **fa** stands for Final Archive, and if the argument is present, all test result data submitted to the database goes directly into the final archive medium (if the argument is omitted, the test data will only be stored on the database server disk)

q if this argument is present, TSCV will command the MTP to start the 'Master AP' and then terminate (if the argument is omitted, TSCV will wait for the MTP to signal that the AP has stopped).

After decoding the command line parameters and determining that the batch mode is requested, TSCV will check that the conditions for starting the test are present. Then it will set–up the test system according to the identified test configuration, without any further user interaction. After completing the set–up, it will command the testnode having the MTP role to invoke an Automated Procedure (AP). One of the attributes in the test configuration end item is MTP_INITIAL_AP, and it is the AP identified by this attribute which TSCV will request TES to invoke.

The MTP_INITIAL_AP must be parameterless!

After starting the AP, TSCV will be waiting for TES to signal that the AP has terminated, unless the optional **–q** option is specified. Then TSCV will terminate immediately after having started the master AP.
When waiting, and TES signals that the AP has terminated, TSCV will also terminate. However, if the optional test session parameter was provided on the command line, it will first close this session.

In the course of checking parameters and checking the start–up conditions, as well as during the set–up and termination, TSCV will send log messages to DBS about the progress and events. If problems are encountered, error messages will also be sent to the CGSI message system.

The Unix process completion code is utilized in the batch mode. Thus, the completion code can be checked to find out if the set–up and test was fulfilled as intended. Completion code 99 represents success. The meaning and implication of most completion codes are rather evident. The event messages from TSCV will aid to trace down the course of the problem more closely.

Two problem situations which may be encountered deserve special description:

1. In the course of the set–up, TSCV will command the MDB software to execute the command **set global default CCU**, where the CCU is the one identified in the command line. TSCV is then obliged to inform the 'participating HCIs' about this event. The set of participating HCI nodes is found from the test configuration end item. If TSCV fails to deliver the message to any of the participating HCIs, TSCV will produce an error message, however, otherwise ignore it.

2. TSCV will ensure that only one instance of the program is active. If, when started in batch mode, it is found that another instance is already running, TSCV will be terminated with the corresponding error code.

7.2.16 The Interactive Mode

All error messages created in interactive mode will be displayed to the user in a pop–up dialog window. The messages may categorized as

- 1 – messages related to the way TSCV is used
- 2 – messages created due to ill–function of TSCV
- 3 – messages resulting from error status returned from the other CGS products with which TSCV interacts.

Copyright per DIN 34

Messages of the first category will in clear text display what the problem is, e.g. that another TSCV instance is already running, or that the user is not authorized to use TSCV.

Messages of the second category represent internal software problems which ideally should have been discovered and removed during implementation and testing. Such errors will be caught in 'exception handlers' in the software, where error messages will be issued. The messages attempts to provide as much information as possible about the condition when the errors occurred. There is little the user can do about such problems, other than report them for maintenance actions.

Messages of the third category attempts to convey to the user the information that the external products provide to TSCV. Simultaneously it is anticipated that the external products logs a message about the problem, and the user should investigate the message log to find out more about the problem and what to do.

Copyright per DIN 34

7.2.17 Batch Mode Exit Codes

The final status after completed execution in batch mode, is indicated by the following exit codes:

- Success 99;
- invalid CCU selection 100;
- invalid mission name 101;
- invalid element configuration 102;
- invalid system tree version 103;
- invalid CCU name 104;
- invalid CCU version 105;
- invalid system tree node name 106;
- failed to connect to other CGS product 107;
- invalid test configuration 108;
- failed to setup test configuration 109;
- failed to start test execution 110;
- failed to create test session 111;
- test configuration already active 112;
- node participating in another active test conf. 113;
- test session name already used 114;
- no final archive device available 115;
- failed to start master AP 116;
- another TSCV instance already invoked 117;
- invalid privilege 118;
- invalid command line profile 119;
- default test session already in use 120;
- TSCV internal error 121;
- maximum number of test configurations 122;
- no replay mode allowed 123;

Copyright per DIN 34

7.3 Test Execution

This section describes how the user having setup the execution session as described in section 8.1 can perform the execution using the CGS provided Human Computer Interface (HCI) product called *Online Control*.

First section 7.3.1 provides an overview of how the user 'sees' the session data and commands the system and how the user can interact with that data during an execution session.

Section 7.3.2 then describes in detail how the user can execute and monitor the session via the HCI Online Control tool.

Appendix K provides some information for advanced users on how the TES behaviour can be controlled by various configuration parameters in a dedicated Configuration File.

7.3.1 Overview

CGS provides a predefined set of "system display" windows to show the status of the equipment under control. It provides dynamic, user controlled display of checkout/control equipment and session data via synoptic displays.

7.3.1.1 Visualisation of Data

The following windows and displays are provided to the user :

USS displays:

– provides mimic diagram of the unit under control/test or control/test equipment
– contains static part (drawing, graphics, bitmaps)
– contains dynamic output elements dynamically animated with data
– contains user interaction widgets (HLCL commands)
– supports up to eight synoptics per computer screen with 50 dynamic outputs each

Message window:

– displays error or other system messages generated by CGS
– type of message can be selected
– source of the message can be selected
– generate audible alarms

Command window

– HLCL commanding of the control/test equipment or a dedicated execution node
– single commands or command sequences
– command sequences from file or from database
– provides command editing capabilites
– provides scrollable command history
– provides possibility to connect to a dedicated execution node

Top level menu("Online Control")

– means to start all HCI windows/applications

System advisory

Copyright per DIN 34

– overall test equipment system status
– overall status of the subsystem

Tree Explorer

– navigation in name tree

Execution node status window

– detailed status of the execution nodes
– acquisition / monitoring / conditions
– stimuli

AP status window

– detailed status of automated procedures running on an execution node
– current UCL statement
– state of the AP

ADU status window

– status about acquisition data units

DB node status window

– detailed status of database node running DBS

SAS status window

– detailed status of the SAS programs running on en execution node

Raw Data Dump Tool

– display contents of ADUs / Tm packets in raw format
– not predefined but online configurable

Copyright per DIN 34

7.3.1.2 Commanding an Execution Session

CGS provides the full range of control from completely manual to completely automatic using the following:

– HLCL commands and HLCL sequences

– Automated procedures
  – run on the execution nodes
  – are compiled and thus run faster and with deterministic timing
  – three priorities: user low, user high, emergency
  – up to 20 AP in parallel on one execution node (* 32 execution nodes)

– Activate Commands/sequences/APs from Synoptics (via mouse click), from command windows, or as a result of monitoring exceptions.

Use of the High Level Command Language (HLCL)

Interpreted keyboard commands issued from the High Level Command Language (HLCL) provide

- single keyboard command
- sequence of individual keyboard commands

HLCL **sequences** can also be defined by the User (offline in a file) and analysed by the receiving software using the HLCL interpreter. No intermediate compilation is required. The HLCL Interpreter will also access the Mission Database to obtain sequences of HLCL commands.

HLCL allows to call any function of imported **UCL user and system libraries**. In particular, HLCL supports the **invocation of AP**s on any execution node, thus establishing a further level of automation in the system as well as interactive access to each execution node.

HLCL also allows to **control the CSS** Simulation Function. Several Commands are available to setup the simulation and to set or read simulation parameters.

Refer to Appendix H for a specification of HLCL Commands provided in a Standard CGS Configuration.

HLCL sequence can be executed in a **single step** mode, where the next statement of the sequence is executed only after confirmation by the user. Refer to Appendix H and chapter 8.3.2 of this document.

Copyright per DIN 34

7.3.1.3 System Housekeeping Data

CGS provides the following system housekeeping data:
- Mode
- Database load point
- free disc space
- printer status
- Actual time
- etc.

It also provides:
- SAS Status
- AP Status
- ADU Status
- Monitoring status
- Stimulus generation status

The values are used in CGS system displays and may be made accessible via SW Variables to UCL / HLCL.

7.3.1.4 Storing of On–line Data

With CGS/ the user can archive the following:
- raw data packets read from SAS
- data packets (stimuli) sent to SAS
- events needed for replay (starting / stopping SMT)

The archived data are used for data evaluation and in replay mode.

CGS also allows for logging of the following:
- error situations
- important events (e.g. sending a stimulus, system status changes)
- ”user events”
- engineering values to be be logged on user request

The user may also establish private data in user private files.

All such data is stored in the result database.

7.3.1.5 Access to Stored On–line Data

During on-line execution the normal evaluation tools can be started to examine the data stored in the result database. Data is really written to result database ”immediately”. The only prerequisite is that the workstation on which the evaluation shall occur must have visibility (UNIX, NFS, ORACLE) to the database. Even the same workstation that is used for execution control can be used for evaluation.

Copyright per DIN 34

7.3.1.6 Automatic Data Supervision Features

Monitoring is the automatic supervision of engineering data and the initiation of predefined actions in case of anomalies detected.

Specific data can be checked against predefined limits such as:
- upper/lower limit
- delta limit check
- danger limit/normal limits
- up to five different normal limit sets with different upper/lower or actions

In case of limit violation the following can take place:
- generate a system error message
- generate a user error message
- issue a command
- issue a sequence of commands
- start emergency automated procedure

For automatic control of data processing functions and applied limit sets **conditions** may be defined in the MDB or online via UCL.

APs written in UCL may be used monitor values and write to software variables, which, in turn are monitored or evaluated by other APs/SAS.

Furthermore, SAS written in Ada may be developped implementing complex data monitoring functions, providing information on out of limit situations as messages or as values written to software variables.

See chapter 8.3.3 for further description.

7.3.1.7 On–line Modifications of the Execution Configuration

CGS allows for the on–line modification (i.e. by updating the items in the MDB and directly using the updated definitions) of the following;

- automated procedures
- HLCL sequences
- synoptics

CGS does not allow on–line modification of measurement and stimulus descriptions in the loaded configuration, but the user can change data limits online as well as applied limit sets. Also, some routing information for ADUs and GDU/SWOP_COMMANDS may be overwritten online, and conditions may be created or deleted.

Copyright per DIN 34

7.3.2 Online Control Applications

The Human Computer Interface (HCI) software provides application/services to control and/or observe the actual running test. In the following the HCI product is called *Online Test Control* because it will not provide the human computer interface to the whole Common Ground Software (CGS), but only to the Test Execution Software (TES).

General aspects and guidelines for the human computer interface (e.g. handling of menus, window layouts) are derived from the HCI Standards based on the OpenLook Styles Guide.

Online Control provides different services and applications to control and monitor the test execution:

- Main Menu
  The main menu starts the Online Control applications/services as described below.
- Screen Setup Service
  The screen setup service enables the user to store and load actual screen setups.
- ADU Status Display
  The ADU Status Display shows the status of all acquisition data units.
- AP Status Display
  The AP Status Display shows the operator a list of the AP's for one test node provided with status information.
- Online Raw Data Dump
  The Online Raw Data Dump Window enables the human user to make a 'snapshot' of current raw data and display it in specific format (hexadecimal/decimal) on the screen.
- SAS Status Display
  The SAS Status Display displays the status (name, service announced, number of error messages, last error message, link identifier) of all special application software running on a test node.
- Execution Node Status Display
  The Execution Node Status Display displays overall status information about an execution node.
- Database Node Status Display
  The Database Node Status Display displays overall status information about the database server node.
- System Advisory
  The system advisory shows the operator the overall status of the database server node, the test nodes, and each sub–system tested. These symbols can be opened like icons to show the operator a synoptic picture of the sub–system/node.
- Command Facility
  The command facility enables the operator to control and monitor test sessions and the test system by entering HLCL commands.
- Explorer
  The Explorer allows to browse to the mission database and to drag'n drop items to HLCL Command Window and Monitoring Window.
- AP Input Dialog
  The AP Input Dialogs are provided to prompt for user input during automated procedure execution.

Copyright per DIN 34

- Out of Limit Display
  Displays all end items of a test node that are currently out of limits.
- Monitoring Window
  Provides an alphanumeric display of selected end items with engineering value, raw value, and detailed status information.

7.3.2.1 Invoking Online Control (HCI)

Online Control is invoked by the CGS Start Center as described in chapter 4.2.

Preconditions for executing “Online Control”:

– All basic CGS processes are running.

– A configuration has been defined and selected by TSCV tool.

– An execution session has defined and selected by the TSCV tool.

– The selected configuration has been started (with all selected execution nodes executing) by the TSCV tool. The TES software runs on all selected nodes.

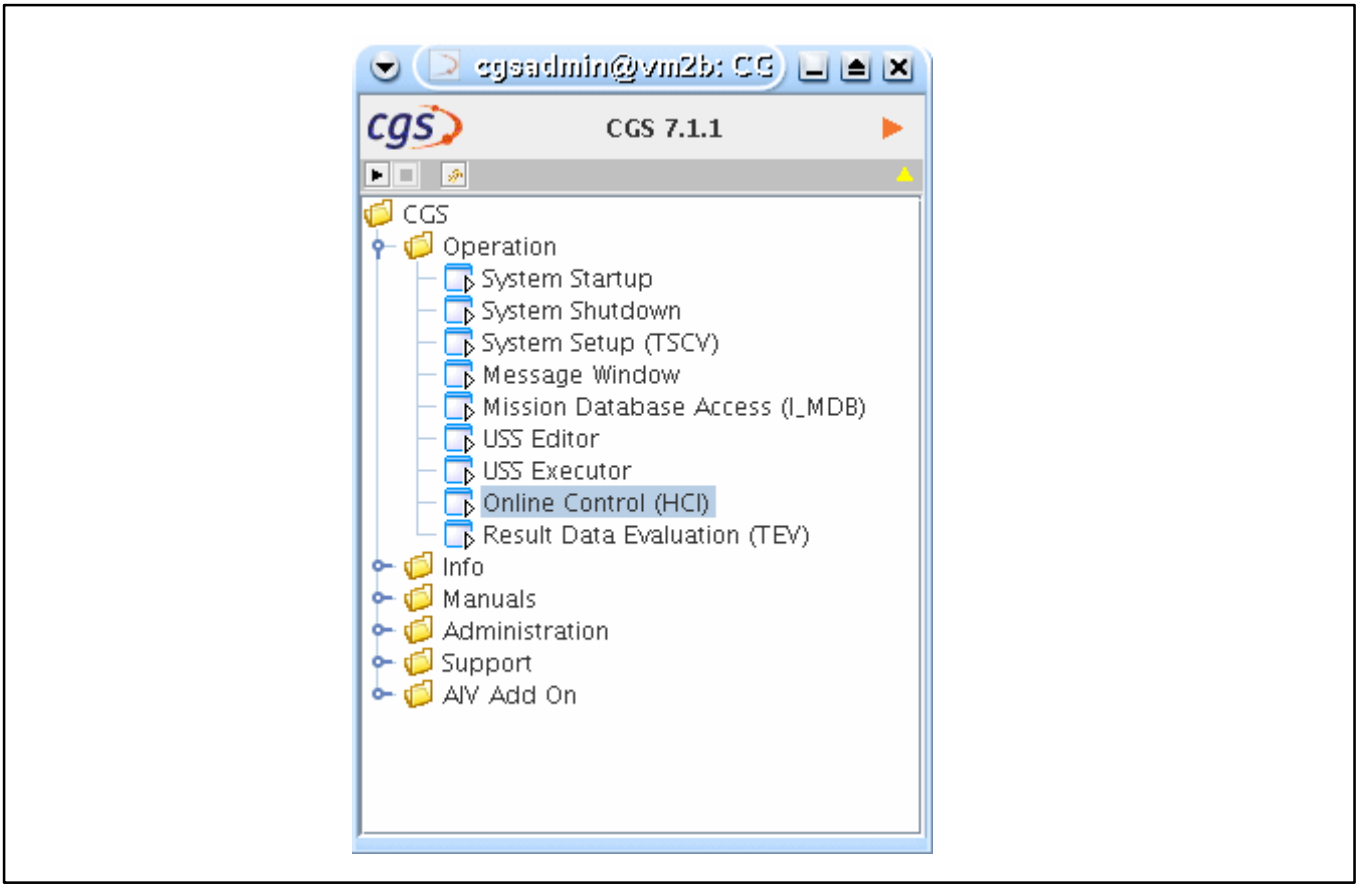


Figure 7–19 : *Invocation of Online Control with Example CGS Start Center*

When going into operation, Online Control displays its main window centered on the screen. On the main window’s status bar (left side of the window’s footer), you may follow the progress of Online Control going in operation. The main window might be repositioned if defined by a default screen setup.

Copyright per DIN 34

When ready the Application menu is enabled and all applications and services are selectable. It displays the node name of the HCI instance and the selected configuration control unit (CCU).

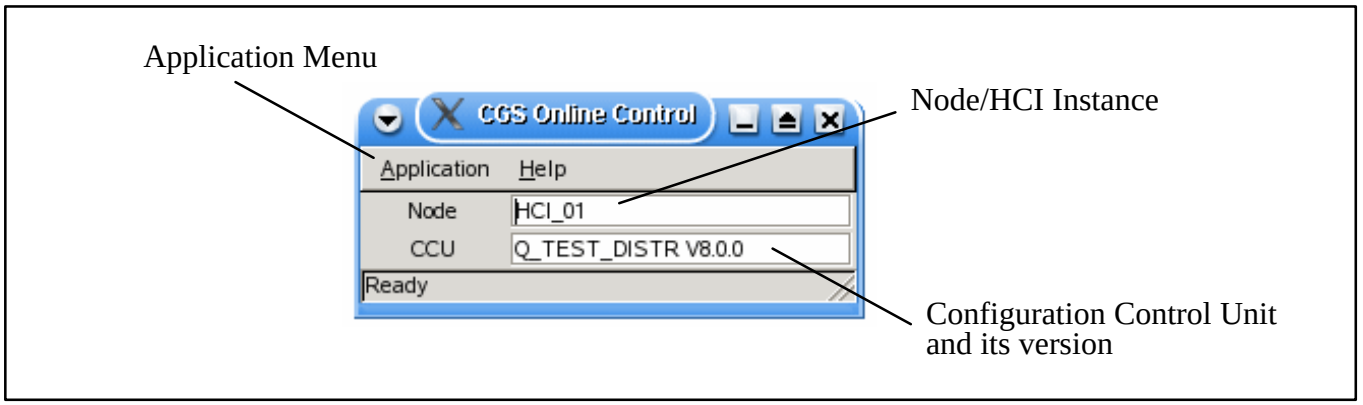


Figure 7–20 : *Online Control (HCI)*

From the main window all Online Control applications are accessible using the Application menu. The application menu is integrated in all Online Control windows having a menu bar.

Copyright per DIN 34

7.3.2.2 HLCL Commanding

When selecting *HLCL Commanding* from the *Application* menu the HLCL Command Facility will be started. If the configuration contains more than one execution node, the default node can be selected by the execution node submenu. All remote commands (e.g. UCL library routines like suspend_ap) will be routed to the default node if no execution node has been explicitly selected.

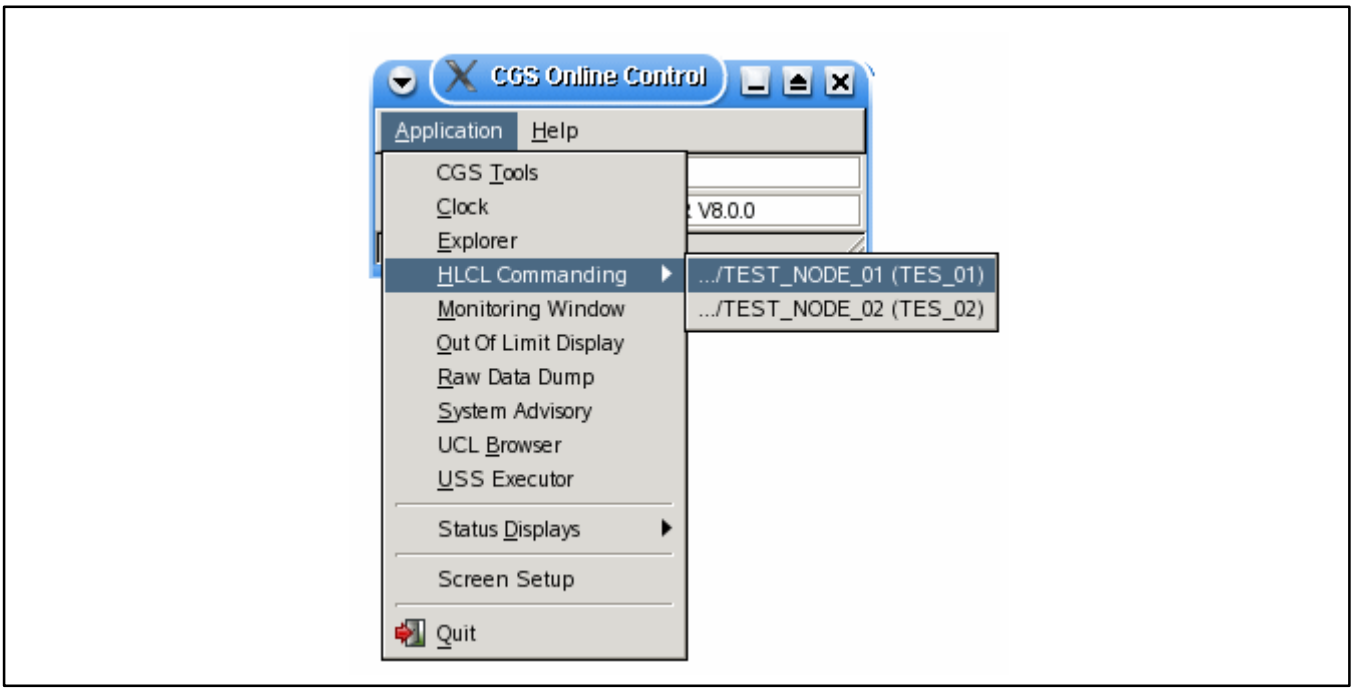


Figure 7–21 *: Execution Nodes Submenu Selection*

Invoking an HLCL Command Tool

- Select **Application –> *HLCL Commanding***

The command window appears on the screen. It consists of an input area and a scrollbar (see Figure 7–22). The command window is ready to accept command input if it shows a new prompt ”HLCL:” with the text cursor symbol behind it.

Copyright per DIN 34

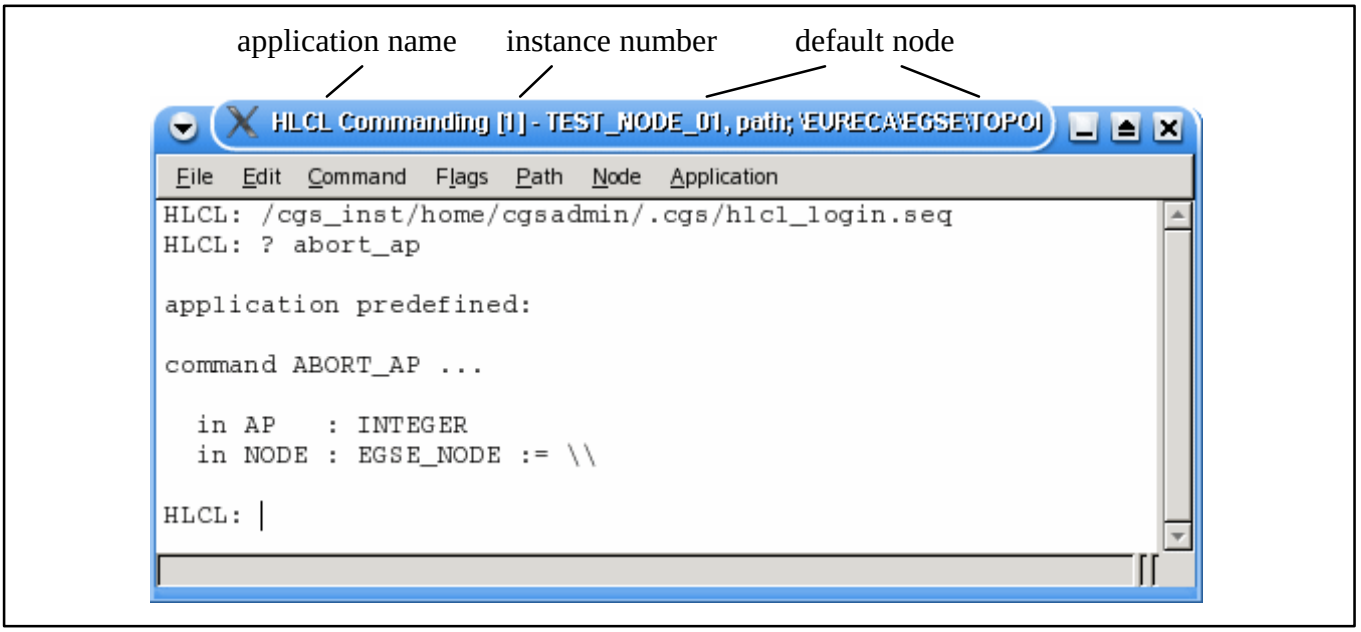


Figure 7–22 *: HLCL Commanding*

To support the command input the following text edit functions (see below) are provided:

- Text insertion, word wrapping, text deletion, and pasting (but only for the actual command)
- Selection and copying (everywhere in the input area).

A command input is terminated by pressing the <Return> or <Enter> key, that will cause the execution of the command.

To get help enter a question mark (<?> key) followed by the <Return> key.

Command Editing

Use text–editing keyboard and menu commands to write and change plain ASCII text.

To set the editing caret, position the mouse pointer and click SELECT. To move the caret, use the arrow keys on the right keypad.

You can cut, copy, paste, or delete text using the context menu available when pressing the right mouse button on the text area. Another method is to select text and press the middle mouse button at the insertion point.

Additional to text edit functions the command window provides a command history function to recall previous entered commands.

Command History

- Pressing the <cursor up> arrow retrieves the predecessor of the displayed command.
- Pressing the <cursor down> retrieves the successor of the displayed command.
- To clear the command line press the ESC–key or CTRL C
- or select ***Command/Interrupt*** from the menu bar.
- When selecting ***Command/History*** from the menu bar a command history window appears as shown in Figure 7–23.

Copyright per DIN 34

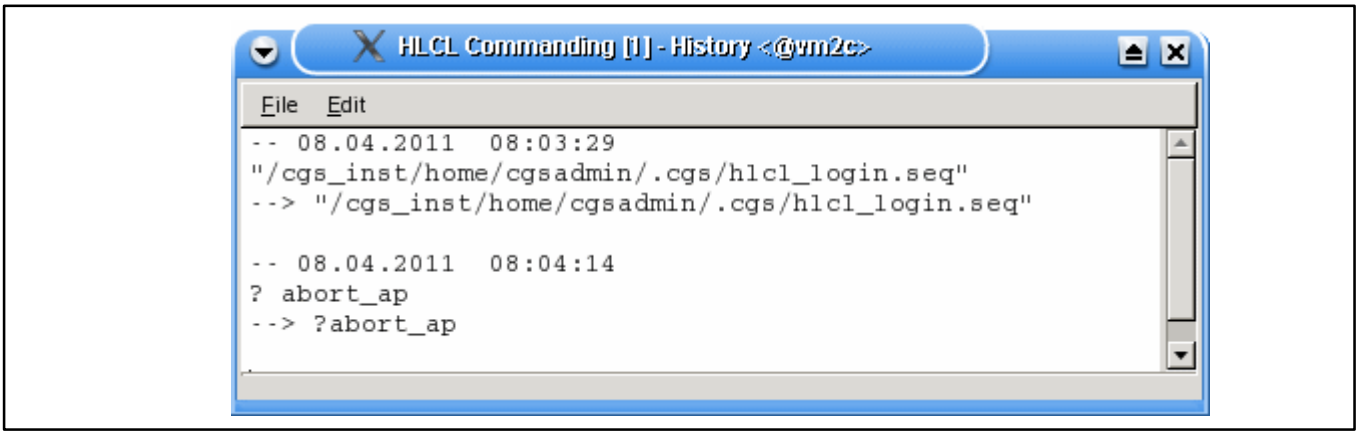


Figure 7–23 *: History Window*

The cursor key history actions will delete the text after the prompt in the command window and display the command from the history instead of it. If the top (bottom) of the history buffer is reached no command is displayed. To reenter the command list press the <cursor down> (<cursor up>) key.
It is possible to edit the ”recalled” commands afterwards.

The user may select one or several commands from the history window and copy/paste them into the command window to execute them.

To cancel the execution of a command the user has to following possibilities.

Command Cancelling

- Press the ESC–key or CTRL C.
- or select ***Command/Interrupt*** from the menu bar.

Note that only some commands can be cancelled.

To stop the execution of a command sequence the user has to following possibilities.

Suspending a Command Sequence (with implicit cancelling of the currently executing command**)**

- Press the ESC–key or CTRL C.
- or select ***Command/Interrupt*** from the menu bar.

Suspending a Command Sequence (without implicit cancelling of the currently executing command**)**

- *Select* ***Command/Suspend*** from the menu bar.

Note that only one command sequence can be suspended at a time. If a new sequence is executed and suspended there is no more a resume possibility for the previously suspended sequence.

Resuming a Command Sequence

- Enter the HLCL command ”Resume”.
- Or select ***Command/Resume*** from the menu bar.

Copyright per DIN 34

A confirmation request is given for commands which are specified for ”confirmation”.

Command Confirmation

- To execute the command type ”y” or ”Y”, to cancel the command type ”n” or ”N” or simply press <Enter> because the latter is the default.

If the user enters an incomplete command the interpreter will ask for missing mandatory parameters.The missing information is prompted for in a loop until the information could be accepted without errors.

Entering Missing Parameters

- Enter the requested parameter end press <Enter> or <Return>.

To terminate the interpreter’s asking simply

- Press the ESC–key or CTRL C
- or select ***Command/Interrupt*** from the menu bar.

A default node or a default path can be chosen in the following way.

Setting a Default Node

- Enter the HLCL command ”Default_Node := ...”.

If the node has been a default node before it is possible to

- select the node from the ***Node*** menu.

Setting a Default Path

- Enter the HLCL command ”Default_Path := ...”

If the default path has been a default path before it is possible to

- select the path from the **Path** menu.

The flag menu is one that eases the setting of some global variables

Invoking the Flags Menu

- When selecting ***Flags*** from the menu bar the flags menu appears on the screen as shown in Figure 7–24.

Copyright per DIN 34

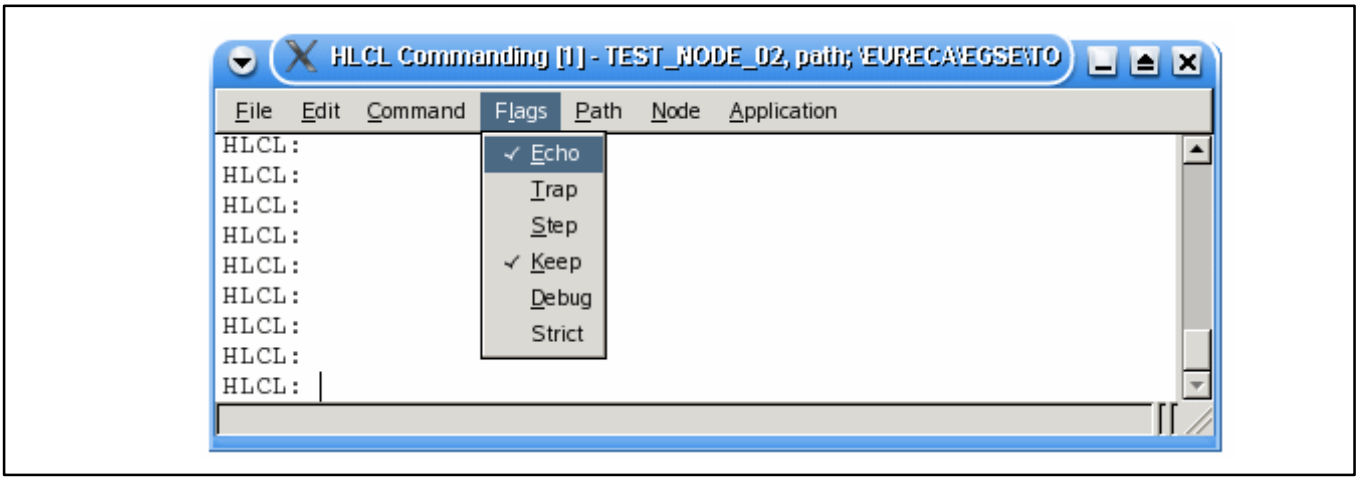


Figure 7–24 *: Flags Menu*

The flags menu has check box items to toggle the state of the HLCL flags. They are used to control command sequence execution and the logging of commands into special command log files.

File Menu

- Clear
  Clear content of the HLCL Command window.
- Open
  Open a file dialog for easy selection of an HLCL command sequence to prepare its execution.
- Save As
  Save the content of the HLCL Command window to a given file.
- Close
  Close the HLCL Command window

To find text in the HLCL Commanding window select **Edit/Find** from the menu bar. This will open a small dialog where the pattern to be searched can be entered. The Next/Previous buttons navigate to the text matching the pattern (case sensitive).

Copyright per DIN 34

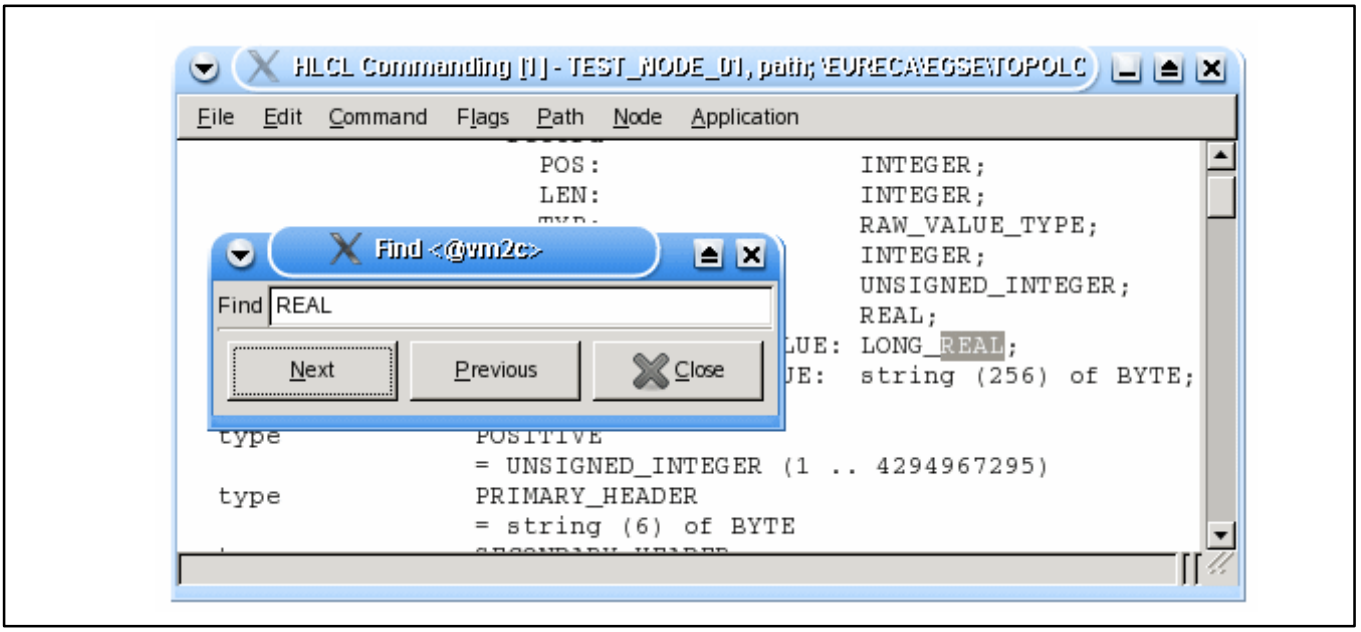


Figure 7–25 *: Find Dialog*

For more information about HLCL commanding refer to Appendix H of this document.

HLCL Login Sequence

Whenever a HLCL Interpreter is created, Online Control searches for a file called hlcl_login.seq in the user's directory $HOME/.cgs. This HLCL sequence file is then executed immediately after creation of a command facility. It's purpose is to execute some general HLCL command like alias definitions or library imports.

The location of the HLCL login sequence can be configured by setting the attribute Online_Test_Control.HLCL.LoginSequence using the CGS configuration editor.

Example: The CGS users shall share the same login sequence (e.g. $CGS_HOME/etc/project/login.hlcl) instead of using an individual one.

1) Open CGS/Administration/Configuration/Global Configuration from CGS Start Center.

2) Search for the group Online_Test_Control.HLCL or create it if not defined.

3) Add the following attribute in HLCL group

LoginSequence = "$CGS_HOME/etc/project/login.hlcl"

4) Save the configuration and start Online Control (HCI).

Configuration

The length of history can be modified bye changing parameter ONLINE_TEST_CONTROL.COMMAND_FACILITY.HISTORY in the CGS configuration (see also Appendix K), default is 200 commands.

The build–in print command can be configured by Online_Test_Control.Printer.LASER_PRINTER1 and Online_Test_Control.Printer.LASER_PRINTER1.

Copyright per DIN 34

7.3.2.3 CGS Tools

To call the CGS Tools submenu, select
Application –> CGS Tools

Note that CGS Tools is avialable via the Application Menu only if enabled by configuration property "Online_Test_Control.CGS_Tools.ShowMenuEntry".

Complementing GUI tools are available, based on a Java Swing platform. These interfaces are:

- Monitoring display interface
- Out–of–limit display interface
- Parameter scrolling interface

A Swing–based MDB tree browser allows drag–and–drop interaction to select items for the monitoring display and parameter scrolling interface. Filtering for out–of–limit displays are supported by MDB subtree selection or limit type selection.

Copyright per DIN 34

7.3.2.4 Data Displays for Online Control

Online Control provides several applications to display data like measurements and housekeeping data.

7.3.2.4.1 ADU Status

To call a ADU Status window, select
Application –> Status Displays –> ADU Status

The ADU Status window (see Figure 7–26) displays all acquisition data units acquired by all execution nodes. It provides the nickname, acquisition status, archiving status, description, and pathname of each acquired ADU.

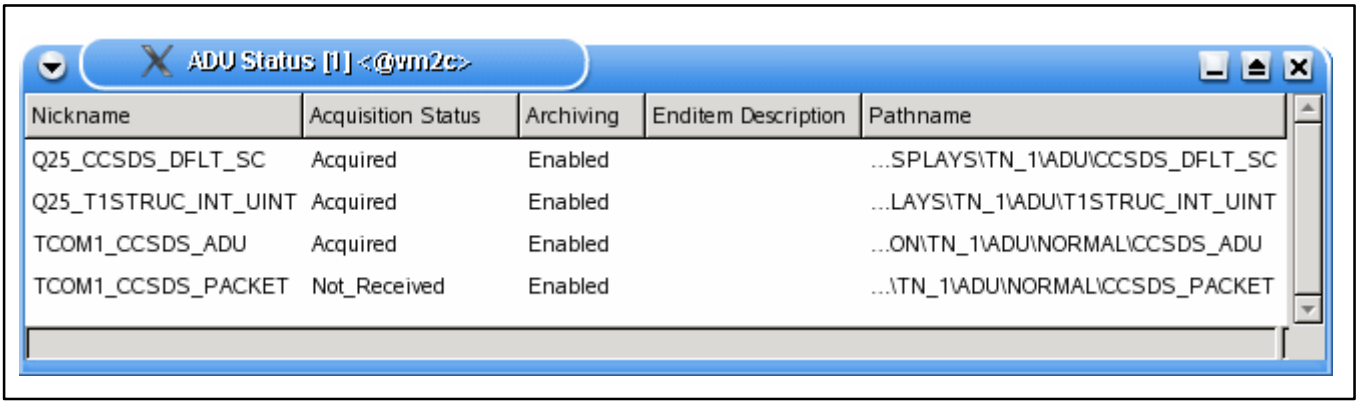


Figure 7–26 *: ADU Status Window*

7.3.2.4.2 AP Status

To call a AP Status window, select
Application –> Status Displays –> AP Status

The AP Status window (see Figure 7–27) shows an overview of the currently executed APs of one specific execution node. If the configuration contains more than one execution node then the node must be selected using the Execution Nodes Submenu. The name of the execution node is displayed on the window header. The AP information is displayed in a scrollable list. The APs are sorted according to their identifier (Id). The number of APs displayed are dependent on the vertical size of the window and are adopted when the window is resized. If the window size is modified in horizontal direction the AP Name is truncated or expanded beginning at the front of the name.

The following items are shown on the AP list:

- AP_Name Pathname of the Automated Procedure. If the pathname is too long to fit on the name field it is truncated on the left side. This is indicated by leading dots.
- Status Actual Status of the AP

NOT_RUN	AP is not running
INITIAL	AP is loaded, but not executed yet
RUNNING	AP is executed
SUSPEND	AP is suspended
TERMINATE	AP is terminated and removed from execution

Copyright per DIN 34

- Id The identifier (number) assigned by TES to each active AP
- Origin Origin from which the AP or parent AP was started.
- Statement Number of UCL statement currently executed.

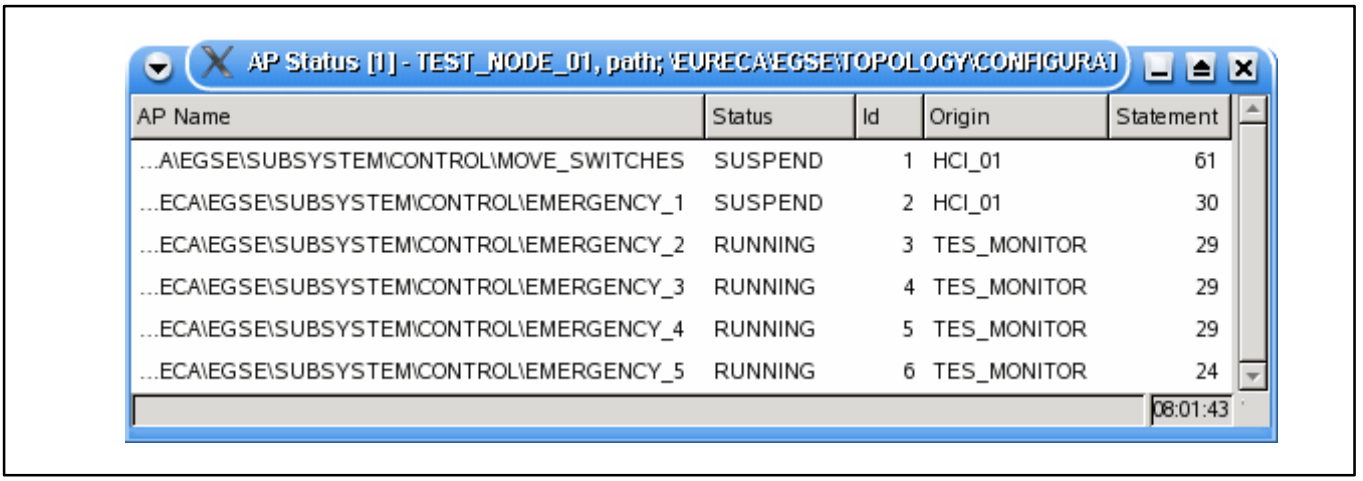


Figure 7–27 : *AP Status Window*

The AP Status window is updated every 5 seconds. The time stamp of the status data is displayed on the right side of the status bar.

7.3.2.4.3 Clock

To call the Clock, select
Application–> Clock

The clock window can be started in two modes, normal and replay mode. In normal mode, the clock window shows the actual time and date of the local time and the simulated mission time (see Figure 7–28). The time is displayed in hh:mm:ss format, the indicator LT is given for local time, the indicator SMT for simulated mission time. The date is displayed in dd.mm.yyyy format. If a time is not available then the digits are replaced by hyphens. Figure 7–29 shows a clock that can't display the simulated mission time.

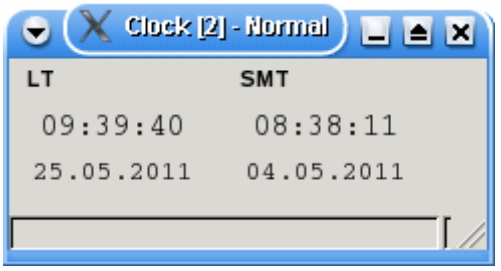


Figure 7–28 : *Clock*

Copyright per DIN 34

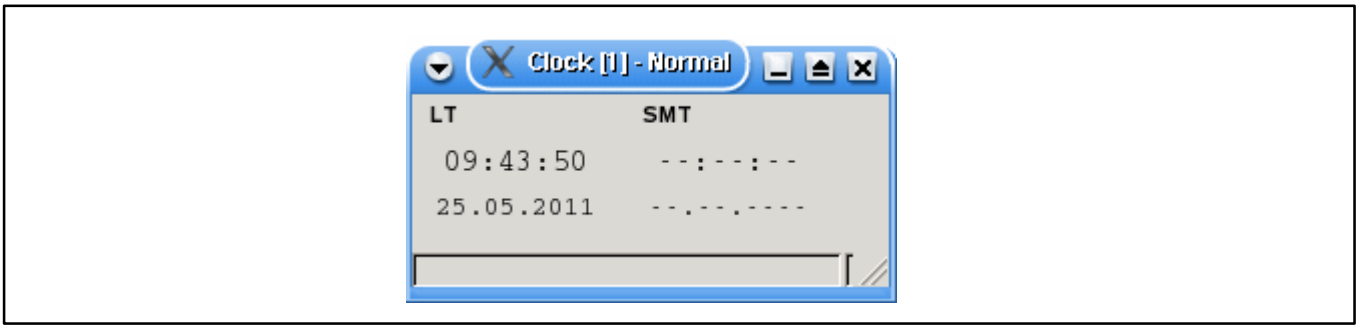


Figure 7–29 : *Clock (SMT not available)*

In replay mode, the clock window displays the recorded local time and simulated mission time as read from archived data by the master test processor. The footer shows the actual time when the data was submitted by the execution node.

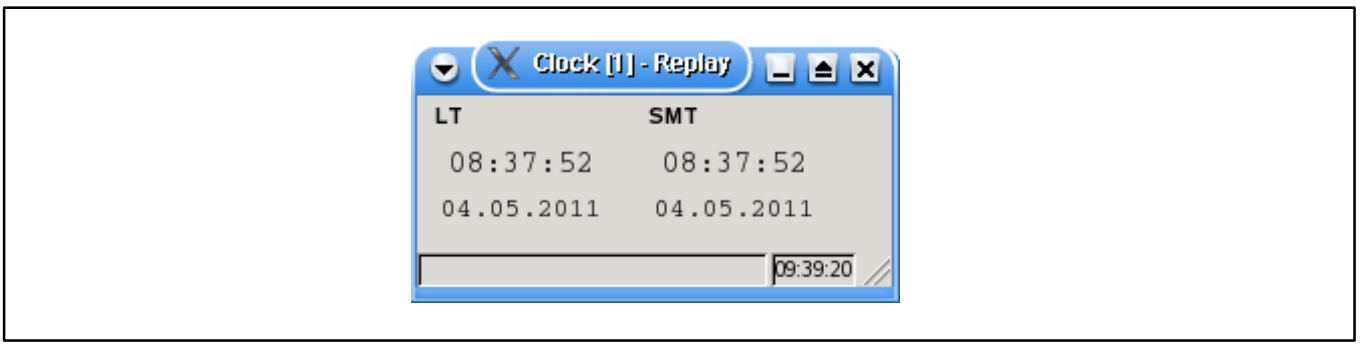


Figure 7–30 : *Clock in Replay Mode*

Copyright per DIN 34

7.3.2.4.4 Database Node Status

To call the Database Node Status window, select
Application–> Status *Displays –> Database Node Status*

The Database Node Status window (see Figure 7–31) displays status information about the CGS Database Server Node. The status data are only available if the Master Test Processor is up and at least in idle mode.

The following status information is provided:

- Session Name
  The name of the actual execution session.
- Free Space (Kbytes)
  The free disc space on TRDB disk ($VICOS_CEN_DBS_HOME) in kilo bytes.
- Connected Evaluations
  The number of evaluation users connected to the test result database.
- DBS Overall Status
  The overall status of the database server node.
  Values are OK, NOT_OK, OTHERS.
- Command History Status
  Status of the command history. Values are OK, NOT_OK. (Here: CGS wasn't configured to provide Command History services)
- TRDB Table Space
  Space (in percent) occupied within the different test result database tables.

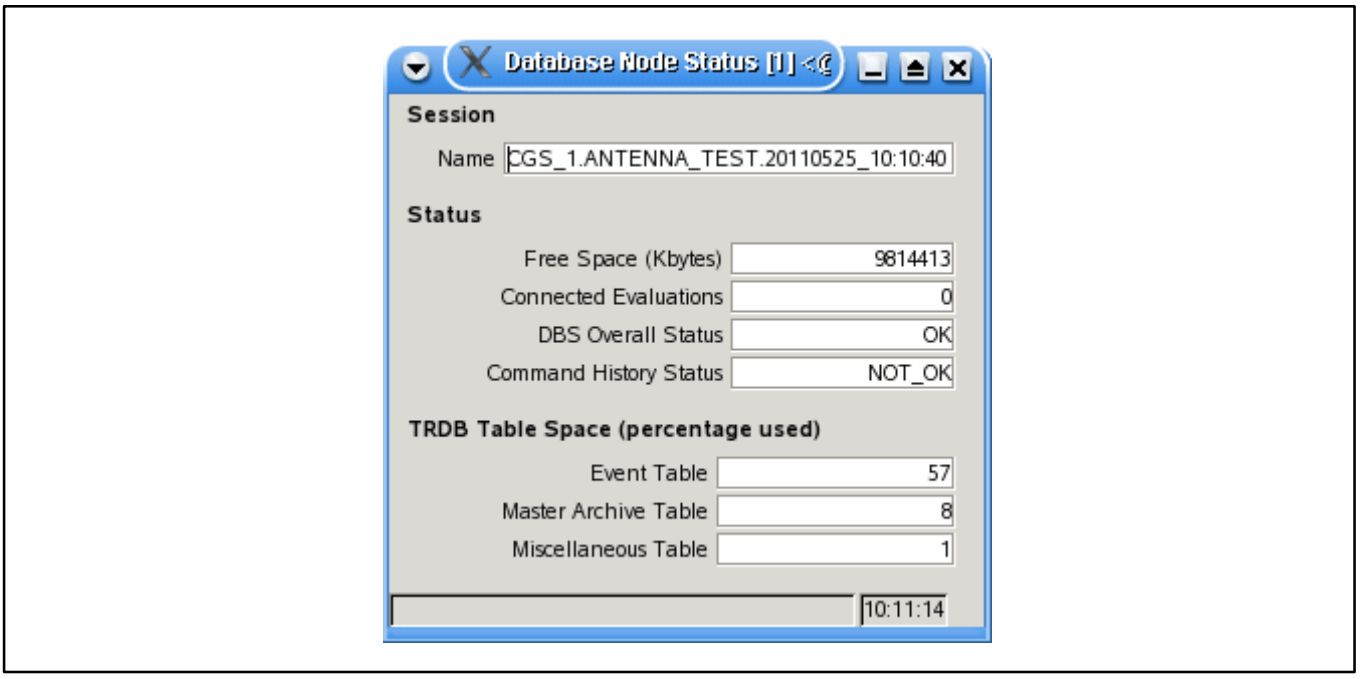


Figure 7–31 *: Database Node Status Window*

Copyright per DIN 34

7.3.2.4.5 Out of Limit Display

To call the Out of Limit Display select
Application–> Out Of Limit Display

The Out of Limit Display shows the enditems of all execution nodes that are actually out of limit.

The total number of items that are out of limit is displayed on the upper left area. Next to this field is the number of the currently displayed items. The numbers may be different if items have been deleted from the table.

The Out of Limit Display can run in two modes: In the keep mode item remain in the table when their status is in limits again. In auto mode items are deleted from the table when they are no longer out of limit.

The following information of an enditem subdivided into 27 columns are available for the user:

1 Acquisition Status
2 Alarm Count
3 Danger Delta (limit)
4 Danger High (limit)
5 Danger Low (limit)
6 Delta Monitoring Status
7 Enditem Description
8 Enditem Name
9 Engineering Value
10 Keep (flag indicating whether the item is kept when going in limits again)
11 Limit Set
12 Max Alarm Count
13 Monitoring Status
14 Nickname
15 Nominal Delta (limit)
16 Nominal High (limit)
17 Nominal Low (limit)
18 OOL Engineering Value (the engineering value when the item went out of limit)
19 OOL Raw Value
20 OOL Time (the time when the item went out of limit)
21 Parameter Number
22 Pathname (the complete pathname of the end item)
23 Processing Status
24 Raw Value
25 Status (the flag)
26 Time Tag
27 Unit

Save and Load of Configurations

If you have a configuration (visibility, order and length of columns) you want to load later again, select *File –> Save Configuration As* and then a file dialog appears. Go to the folder where you want to save the configuration. Write a filename you want to use to the Name text field and press the Save button. If no error message is reported the configuration is saved. All filenames of Out Of Limit Display Configurations have the postfix ".ool" .

Copyright per DIN 34

If you want to load a saved configuration go to the folder where the file is saved, select the file name in the list with a click and then press the Open button. If it is not a valid configuration file a error message is reported else the actual Out Of Limit list is deleted (!) and the new configuration is loaded.

See section 7.3.2.4.7 (Monitoring Window) for color coding and handling of the file dialog.

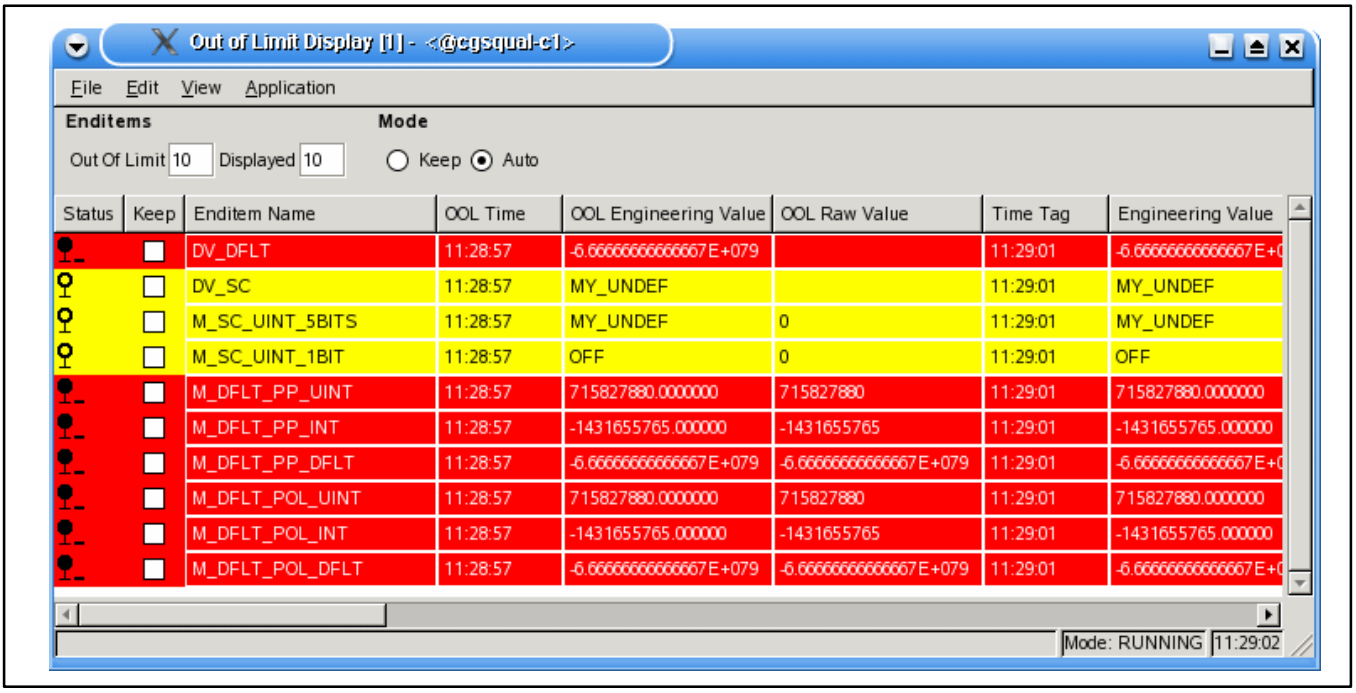


Figure 7–32 : *Out of Limit Display*

Preferences

The size of all columns can be changed: When moving the cursor on the table header to the right column margin, the mouse cursor changes to a resize cursor (<=>). In this mode the column left to the cursor can be resized by moving the mouse left or right while keeping the left mouse button pressed.

To change the order of the columns select a column header and drag it to its new position while holding the left mouse button pressed.

The visibility of the columns can be changed using the properties dialog. To open the properties dialog select *Edit –> Preferences* from the menu bar. The Show Columns area of the preferences dialog has a check box for each column. When checked the column is shown in the table, when unchecked it is hidden.

The raw values can be displayed as binary, decimal or hexadecimal numbers.

Press the Apply button to accept all changes that are made or press the Cancel button to reject them. When pressing Apply As Default, the properties are used as default for all newly created Out Of Limit Displays. When applied as default, the properties are stored in the user's CGS configuration file.

Copyright per DIN 34

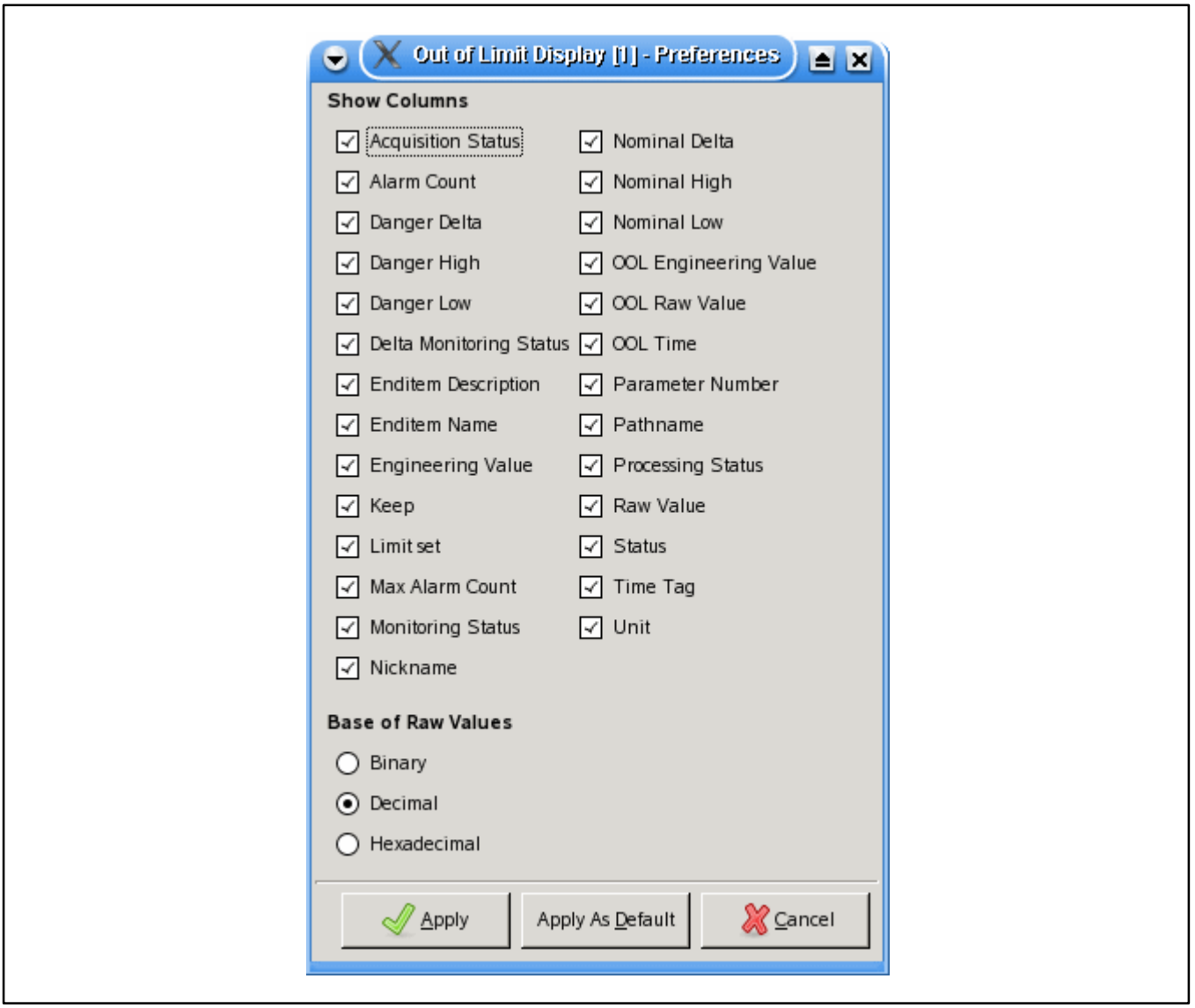


Figure 7–33 : *Out of Limit Display Preferences*

Edit Actions

There are some edit actions defined on displayed items:

- Clear
  Clears the complete content of the Out of Limit Display.
- Delete
  Deletes the selected items from the window until they become “out of limit” again.
- Auto Remove
  The item will be automatically removed from the window if its status changes from ”out of limit” to “in limits”.
- Keep
  Keeps diplaying this item whatever status it has. It is the default behaviour of all new items in the Out Of Limit Display.
- Select All
  Select all items.

Copyright per DIN 34

- Deselect All
  Deselect all items.
- Preferences
  Open the preferences dialog.

Delete, Auto Remove, and Keep are only available if one or more items are selected.

The View menu can be used to show details of a selected item.

Delete, Auto Remove, Keep, and Details are also available from the context menu when pressing the right mouse button on the table area.

Colors and Symbols

According to the actual monitoring and delta monitoring status of the end item the row is signed with a special symbol (status column) and painted in a special color defined in the cgs_configuration.xml file:

(Delta) Monitoring Status	Symbol	Color (default)
UNDEFINED (no limits defined)	no symbol	brown
DISABLED	no symbol	cyan
IN_LIMITS		green
NOMINAL_LIMIT_VIOLATION		yellow
NOMINAL_LOW_LIMIT_VIOLATION		yellow
NOMINAL_HIGH_LIMIT_VIOLATION		yellow
DANGER_LIMIT_VIOLATION		red
DANGER_LOW_LIMIT_VIOLATION		red
DANGER_HIGH_LIMIT_VIOLATION		red

7.3.2.4.6 Graph Facility

The Graph Facility is no longer part of the Online Control software. It was substituted by USS Executor's Quick Graph.

Copyright per DIN 34

7.3.2.4.7 Monitoring Window

To call the Monitoring Window, select
Application –> Monitoring Window

In this window user defined items can be monitored. The item information are displayed in a table.

The following information of an end item subdivided into 22 columns is available for the user:

1 Acquisition Status
2 Alarm Count
3 Danger Delta (limit)
4 Danger High (limit)
5 Danger Low (limit)
6 Delta Monitoring Status
7 Enditem Description
8 Enditem Name
9 Engineering Value
10 Limit Set
11 Max Alarm Count
11 Monitoring Status
12 Nickname
13 Nominal Delta (limit)
14 Nominal High (limit)
15 Nominal Low (limit)
16 Parameter Number
17 Pathname
18 Processing Status
19 Raw Value
20 Status (flag)
21 Time Tag
22 Unit

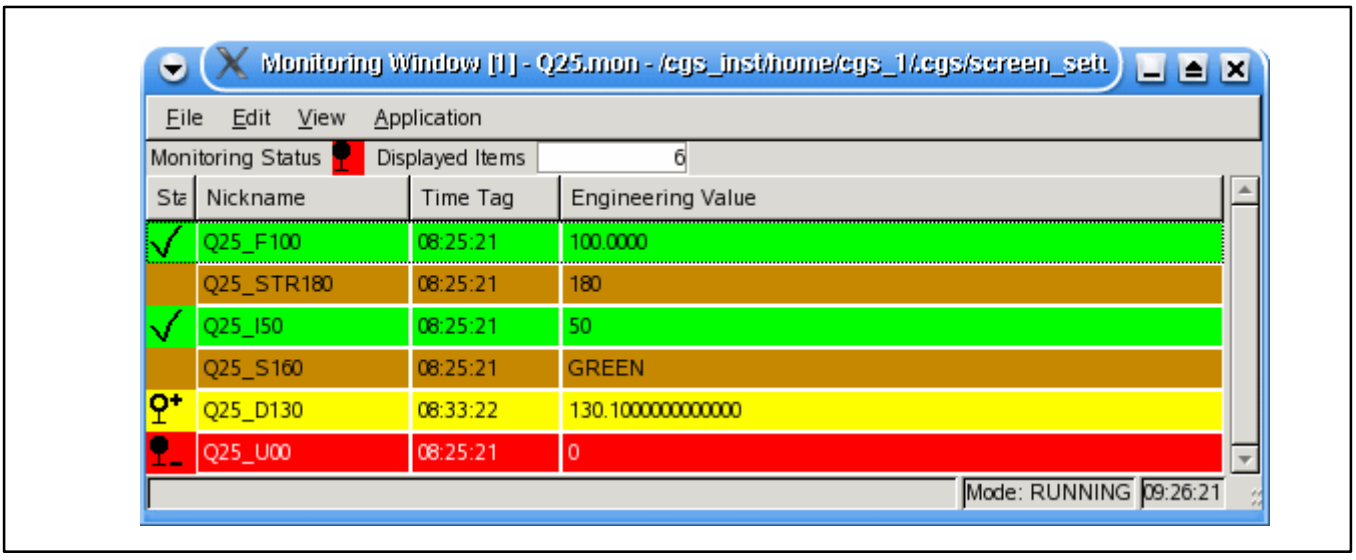


Figure 7–34 : *Monitoring Window*

Copyright per DIN 34

File Menu

If you have a Monitoring Window list of items you want to load later again, select *File –> Save As* and then the Save File As Dialog appears. Go to the folder where you want to save the configuration (the default is $HOME/.cgs/screen_setup_pool). Write a filename you want to use to the Name field and press the Save button. If no error message is reported the Monitoring Window configuration is saved. All filenames of Monitoring Window configurations have the postfix ".mon" . When a configuration has been saved or opened the file name and location is displayed on the window header. Changes can then be directly stored using the *File –> Save* menu.

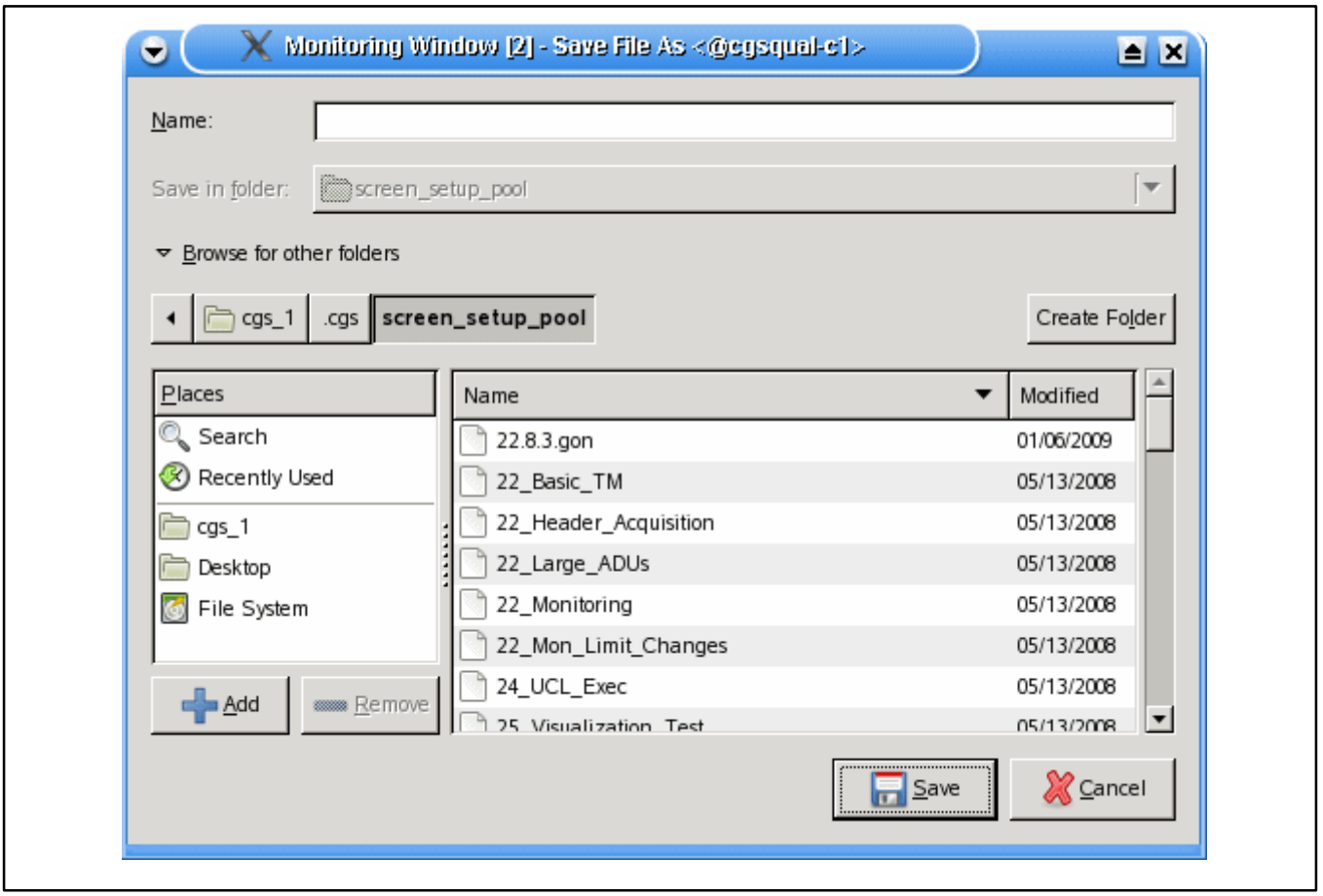


Figure 7–35 *: Monitoring Window Save Configuration*

If you want to load a saved configuration go to the folder where the file is saved, select the file name in the list with a click and then press the Open button. If it is not a valid configuration file a error message is reported else the actual Monitoring Window list is deleted and the new list from the configuration file is loaded.

A configuration can also be loaded by drag 'n drop a file from a file browser like KDE's konqueror to the table area of the monitoring window.

Copyright per DIN 34

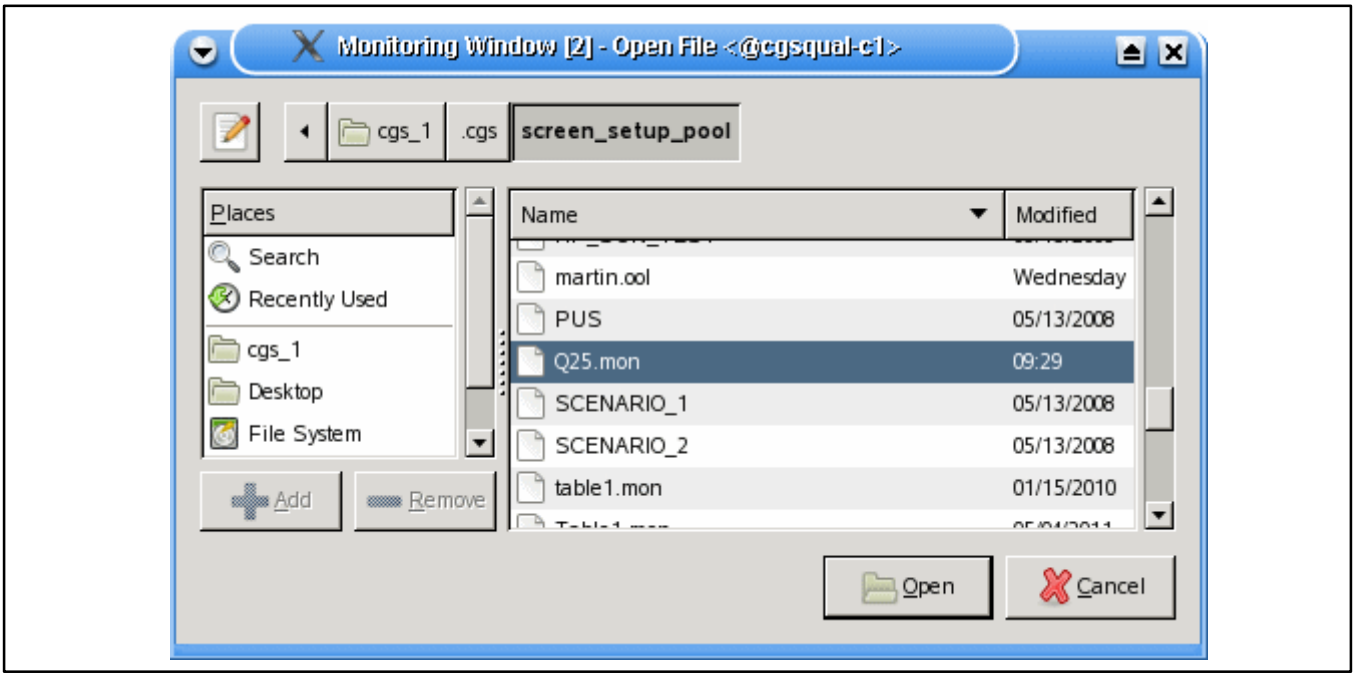
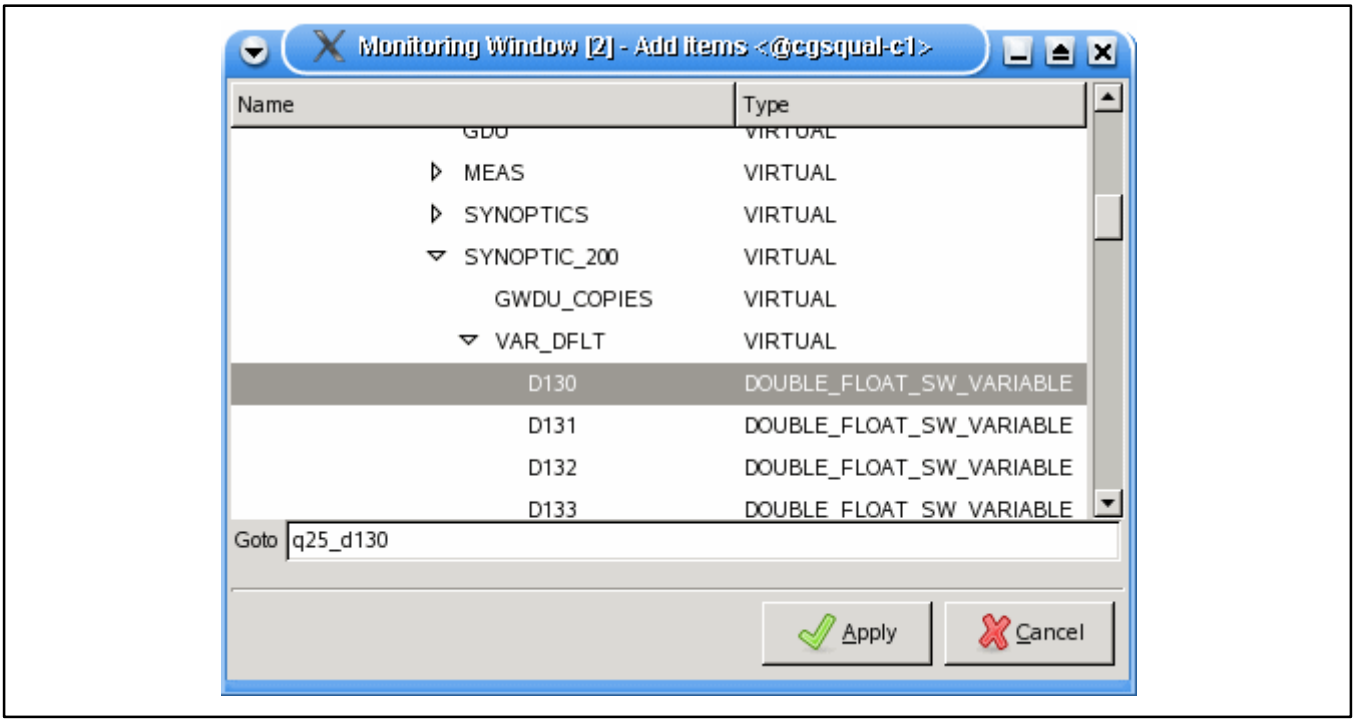


Figure 7–36 *: Monitoring Window Load Configuration*

Often used folders can be bookmarked using the *Add* button or context menu. They will then appears in the Places area. On unix systems file names starting with a dot are treated as hidden files (like ".cgs"). Their visibility can be toggled by the context menu's check item "Show Hidden Files".

Edit Menu

Select the *Edit –> Add* menu to add one or more items to the Monitoring Window. The Add Items dialog pops up where you can navigate through the mission database and add end items to the Monitoring Window list.



Copyright per DIN 34

Figure 7–37 *: Monitoring Window Add Items Dialog*

In the upper area you can browse through the mission database. Folders are marked by small triangles and can be opened by clicking on them. A single item can be selected by pressing the left mouse button. This will remove any previous selection. A range of items can be selected by a left click on the first item and then left clicking the last item while pressing the SHIFT key. Additional items can be added to a selection when pressing the CTRL key when clicking on them. All selected items are added to the monitoring window when pressing the *Apply* button. If a folder is selected all item below that folder are added.

The Goto text input field can be used to quickly select an item in the browse area. When the text matches a pathname, nickname, or short identifier the item is displayed and selected in the browse area. Previous selections are removed.

When a lot of items have been selected a progress bar becomes visible. The process of adding items can be aborted by closing the dialog.

Figure 7–38 *: Monitoring Window" Add in progess...."*

With *Edit –> Clear* you can delete all enditems from the monitoring window table. To delete specific items from the table select them with the left mouse button and choose *Delete* from the Edit or context menu. A range of items or multiple selections can be made using the SHIFT or CTRL key as described for the "Add Items" dialog. *Edit –> Preferences* opens the preferences dialog to configure the column visibility as described for the Out of Limit Display. The column order and size is also configured in the same way.

View Menu

The *Details* menu item of the View menu pops up a text dialog showing the details of the selected item. It is also available by the context menu.

Copyright per DIN 34

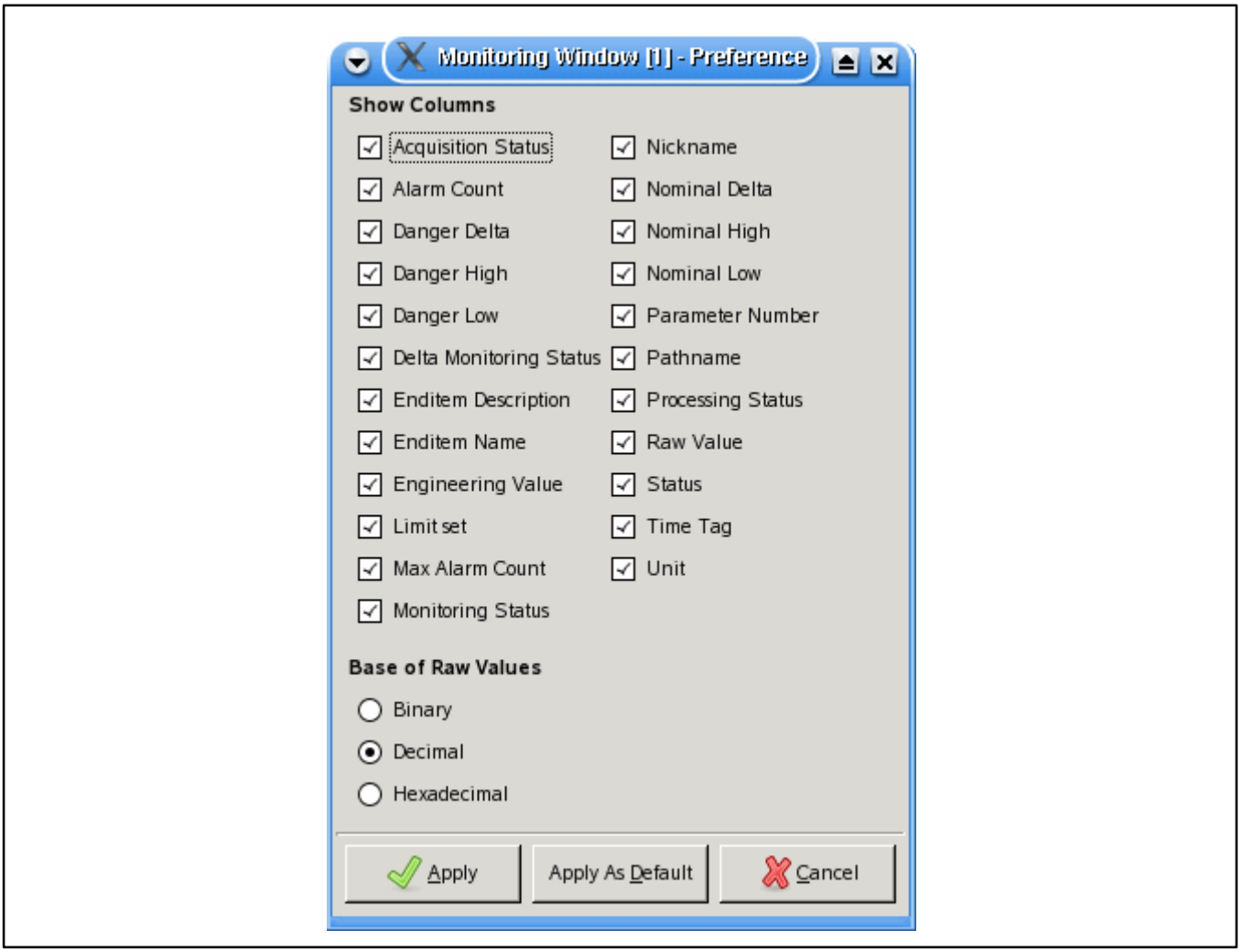


Figure 7–39 : *Monitoring Window Preferences*

Copyright per DIN 34

Symbols

According to the actual monitoring and delta monitoring status of the end item the row is signed with a special symbol and painted in a special color defined in the cgs_configuration.xml file:

(Delta) Monitoring Status	Symbol
UNDEFINED (no limits defined)	no symbol
DISABLED	no symbol
IN_LIMITS	
NOMINAL_LIMIT_VIOLATION	
NOMINAL_LOW_LIMIT_VIOLATION	
NOMINAL_HIGH_LIMIT_VIOLATION	
DANGER_LIMIT_VIOLATION	
DANGER_LOW_LIMIT_VIOLATION	
DANGER_HIGH_LIMIT_VIOLATION	

Copyright per DIN 34

Colors

If processing is disabled for a telemetry item, the default color is cyan. If processing is enable, the color is determined according to this table:

Acquisition Status	**Monitoring Status**	**Delta Monitoring**	**Color (default)**
Requested, Not_Maintained, Not_Acquired, Not_Received, Invalid	any	any	purple
Static	any	any	cyan
Acquired, Data_Interruption	In_Limits, Undefined	Disabled	turquoise
Acquired, Data_Interruption	In_Limits, Undefined	In_Limits	green
Acquired, Data_Interruption	In_Limits, Undefined	Delta_Soft_Limit_Violation (Nominal Limit)	yellow
Acquired, Data_Interruption	In_Limits, Undefined	Delta_Hard_Limit_Violation (Danger Limit)	red
Acquired, Data_Interruption	In_Limits, Undefined	Undefined	brown (rgb:c7/89/00)
Acquired, Data_Interruption	Nominal_Limit_Violation, Nominal_Low_Limit_Violation, Nominal_High_Limit_Violation	Delta_Hard_Limit_Violation	red
Acquired, Data_Interruption	Nominal_Limit_Violation, Nominal_Low_Limit_Violation, Nominal_High_Limit_Violation	else	yellow
Acquired, Data_Interruption	Danger_Low_Limit_Violation, Danger_High_Limit_Violation	any	red
Acquired, Data_Interruption	Disabled	any	turquoise

Copyright per DIN 34

7.3.2.4.8 Raw Data Dump

To call a Raw Data Dump window, select
Application–> Raw Data Dump

The purpose of the Raw Data Dump is to display the content of acquisition data units (ADU).

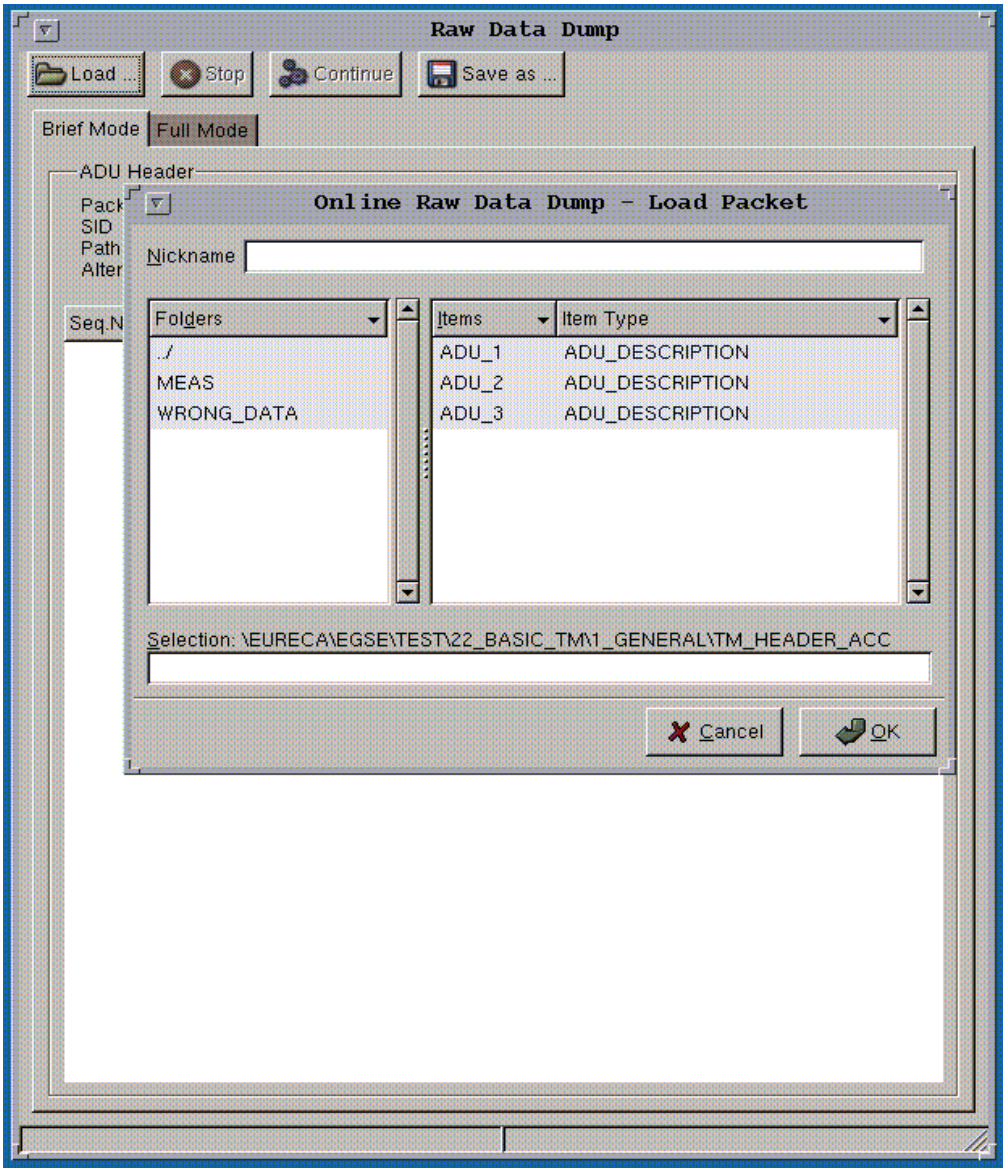


Figure 7–40 *: Raw Data Dump*

To dump an ADU, select the load button and enter the pathname of the ADU in the Load dialog. If the ADU is available, the raw data is displayed according to its type (unstructured, structured, or CCSDS packet) and format. To change the format (not applicable to structured ADU's), select the format button and choose the new format from the dialog.

Figure 7–41 *: Raw Data Dump Window (Load Packet)*

Copyright per DIN 34

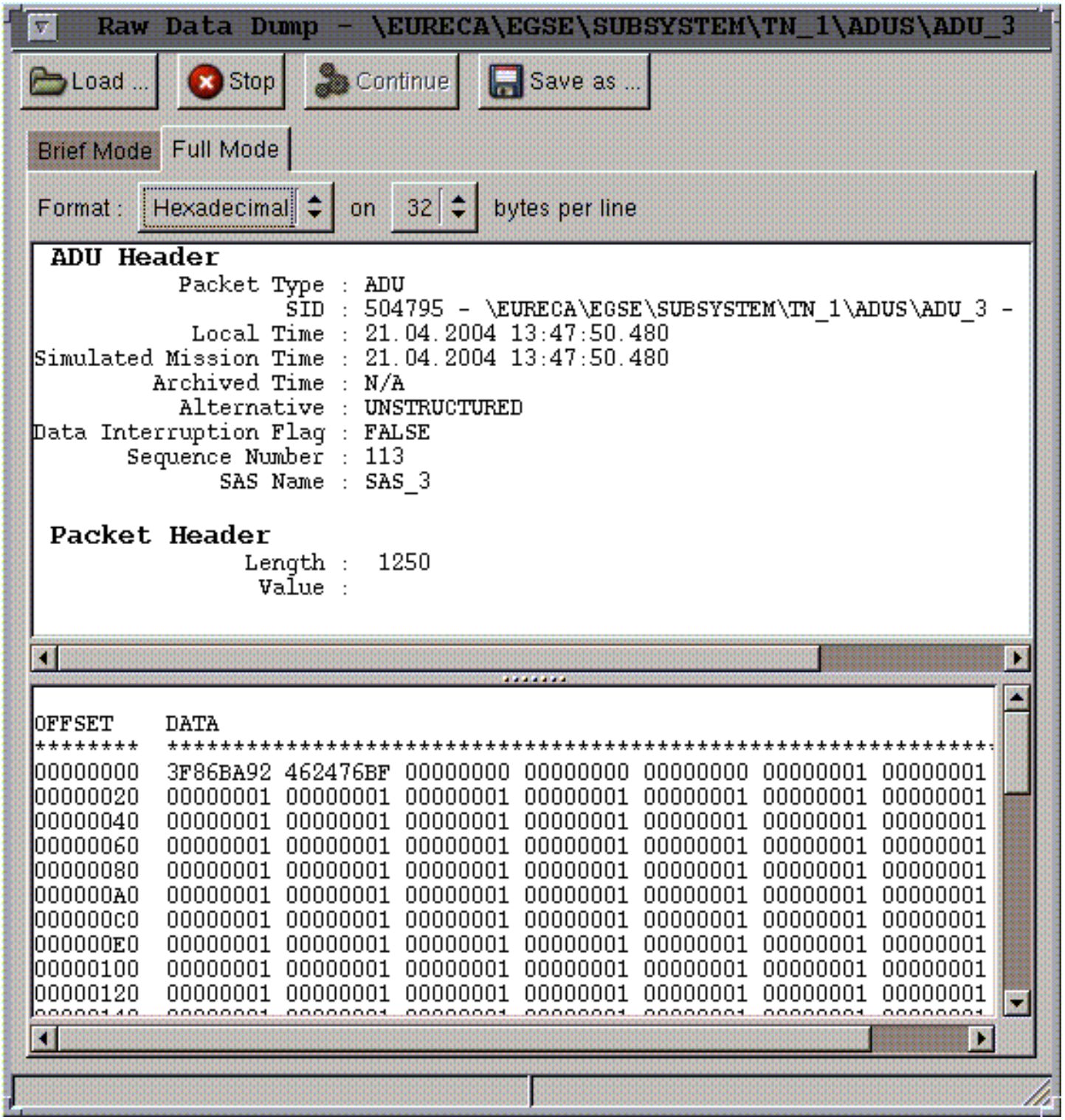


Figure 7–42 : *Raw Data Dump Window (Full Mode)*

Remark: When not monitoring the raw data, stop the data display or quit the Raw Data Dump, because updating text windows consumes a lot of processing power.

Copyright per DIN 34

7.3.2.4.9 SAS Status

To call a SAS Status window, select
Application–> Status Displays –> SAS Status

The SAS Status window shows an overview of the currently executed Specific Application Software (SAS) programs running on a specific execution node. A specific execution node can be selected like for the AP Status application (see 7.3.2.4.2). The name of the execution node is shown on the header/title bar of the SAS Status window.

The following items are shown on the window (see Figure 7–43):

- SAS_Name
  The Unix file name of the SAS
- Service
  The service announced by the SAS:
  – NONE no service announced
  – ADU_SERV the SAS sends ADUs
  – GDU_SERV the SAS accepts GDUs
  – ADU_GDU the SAS sends ADUs and accepts GDUs
- Messages
  The number of messages sent by the SAS to TES
- Last Message
  Text of last message sent by the SAS. This field is truncated according to the width of the SAS Status window.
- Id
  The Application Identifier of the SAS (i.e. a number assigned by TES)

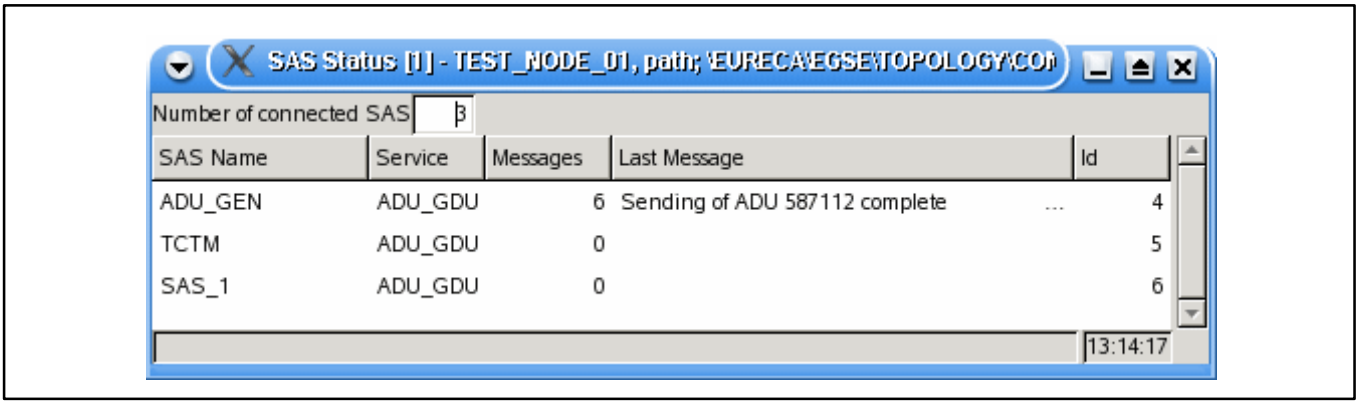


Figure 7–43 *: SAS Status Window*

Abbreviated information can be made visible by clicking on the corresponding item like on the AP Status application (refer to 7.3.2.4.2).

7.3.2.4.10 System Advisory

To call the System Advisory, select
Application–> System Advisory

Copyright per DIN 34

The System Advisory window displays the overall status of the execution nodes, the subsystem/unit under control, and the DB server node (see Figure 7–44).

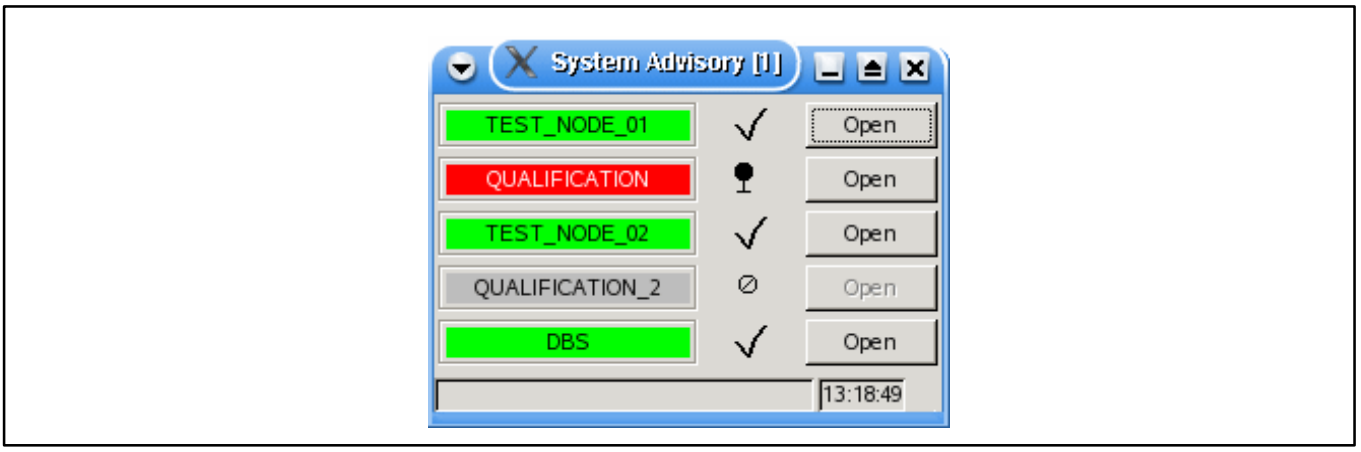


Figure 7–44 *: System Advisory*

The execution node boxes are labelled with their name as defined in the mission database and are color–encoded and symbol–encoded with following status indication:

- green and symbol ✓ :
  The execution node status is ok
- yellow and symbol ⚲ :
  Not applicable.
- red and symbol ⚲ :
  The execution node status is error.
- grey and symbol ⊘ :
  The execution node is not running or doesn't provide data in time.

By selecting the *Open* button right to an execution node box the corresponding Execution Node Status window will be opened.

The subsystem boxes are labelled according to the definition of the EGSE Test Nodes of the EGSE test configuration. To define a label start the I_MDB tool, select an EGSE test node from the test configuration and edit the Subsystem Name attribute (see Figure 7–45).

Remark: Modifications of a configuration (e.g. overview synoptic, subsystem name) are available not until (re)loading the configuration with TSCV.

Copyright per DIN 34

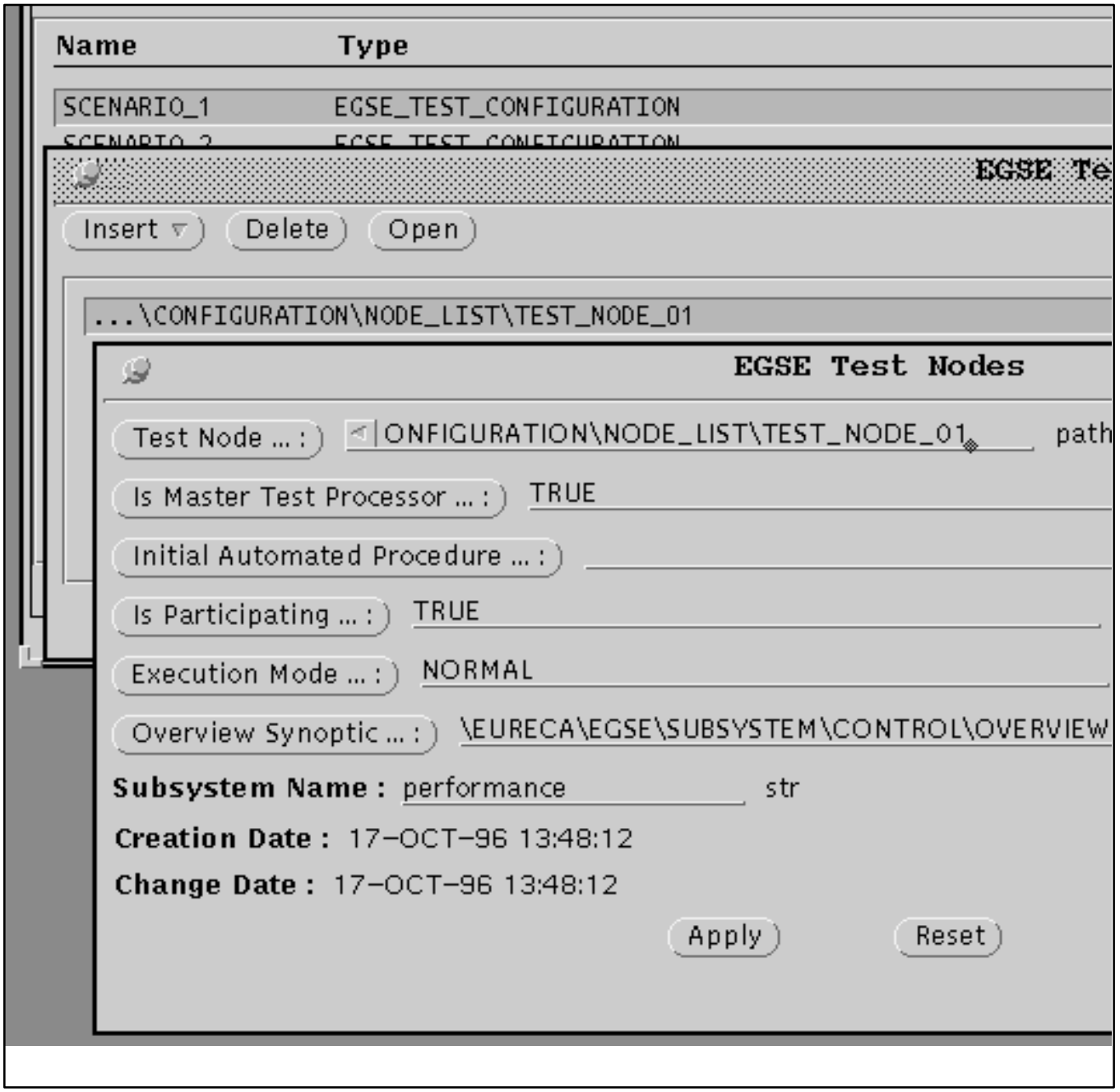


Figure 7–45 *: EGSE Test Nodes Window*

Copyright per DIN 34

For the subsystem boxes the colors and symbols indicate the following status:

- green and symbol ✓ :
  All monitored end items are in limits.
- yellow and symbol :
  At least one end item is out of nominal/soft limit.
- red and symbol :
  At least one end item is out of danger/hard limit.
- grey and symbol ⊘ :
  No end items are enabled for monitoring.

By selecting the *Open* button right to a subsystem box the corresponding overview synoptic will be opened. The synoptic can be specified in the same way like the subsystem name (s.a.).

For the database server (DBS) box the following status are applicable:

- green and symbol ✓ :
  The status of the DBS is ok.
- yellow and symbol :
  The status of the DBS is warning.
- red and symbol :
  The status of the DBS is error.
- grey and symbol ⊘ :
  The DBS status can not be acquired from the master test processor.

By selecting the *Open* button right to the DBS box the Data Base Node Status window will be opened.

Copyright per DIN 34

7.3.2.4.11 Explorer

To call the Explorer, select
Application–> Explorer

The Explorer window displays the name tree of the MDB. Currently, it can only be used as drag'n drop source for the HLCL Command window and Monitoring Window and to copy pathnames to the clipboard. When an item is selected in the Explorer it can be dragged and dropped on the HLCL Command Window. It will then display the pathname of the item in the HLCL Command Window. When dropping an item on the Monitoring Window, the item or all items below a folder will be added. When an item is selected in the Explorer the pathname can be copied to the clipboard by selecting Copy item from the context menu.

The Explorer provides a simple search function in the bottom area of the window. It finds all occurences of the pattern in pathnames and nicknames and shows the number of hits on the footer. The Down and Up buttons can be used to navigate from one hit to the other. Pressing the <Return> key has the same effect as clicking on the Down button.

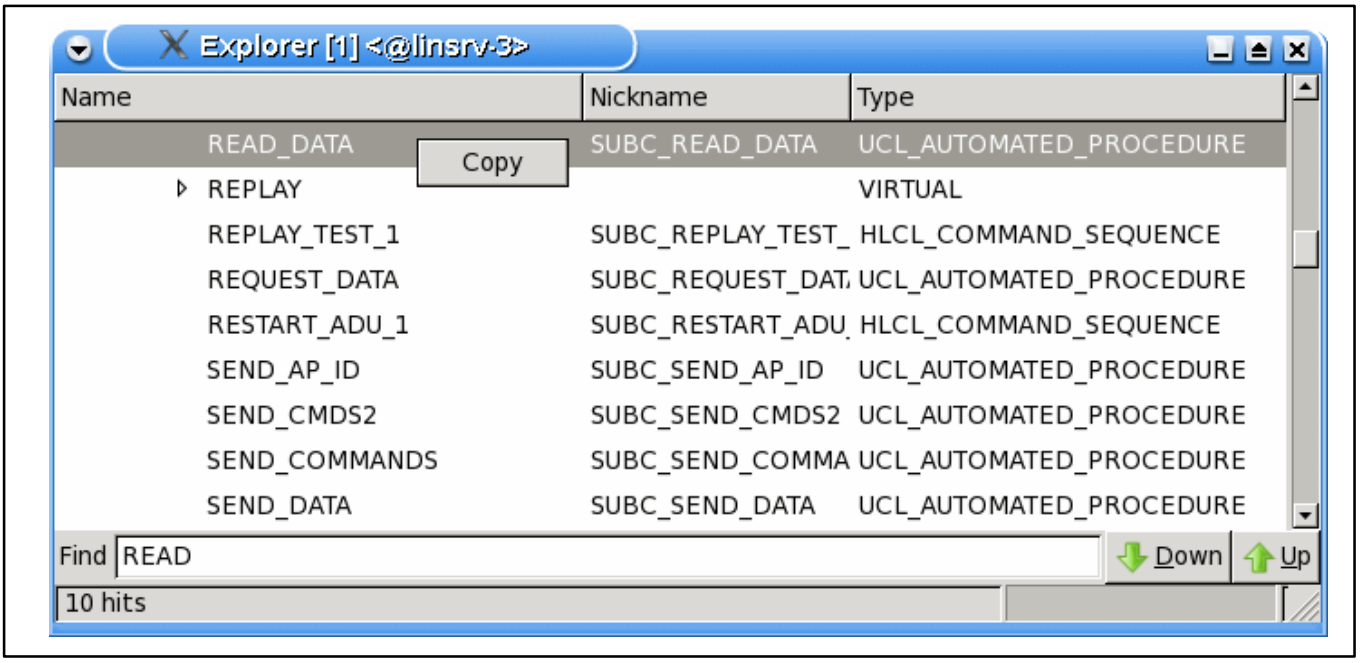


Figure 7–46 *: Explorer Window*

Copyright per DIN 34

7.3.2.4.12 Execution Node Status

To call the Execution Node Status window, select
Application–> Execution Node Status

The Execution Node Status window shows an overview of some housekeeping values of an execution node. The execution node can be selected by the Execution Nodes submenu; the MTP is used as default.

The status information displayed on the Execution Node Status is grouped into six categories. The category currently displayed can be selected via the tabs.

- General (see Figure 7–47 below) tab
  presents general status information
  - Configuration Path
    The value shows the pathname of the configuration used to setup the system.
    It determines which data have been loaded by the execution node, too.
  - MTP
    Indicates whether the test node is configured as Master Test Processor (MTP) or Special Check Out Equipment (SCOE) node.
  - Current Mode
    Current execution mode

NONE	No mode was selected
NORMAL	The execution node is connected to the normal data acquisition and data generation links.
REPLAY	The execution node replays data from previously archived data.
SIMULATE	The execution node simulates data acquisition and data generation links: Measurement data is generated and Telecommands/Stimuli are internally accepted, but not sent.

  - Status
    Current status of the execution node

IDLE	The execution node is idle.
RUNNING	The TES software on the execution node is active and accepts UCL commands / data requests.
SUSPEND	The TES software running on the execution node is suspended. No other commands than a mode switching command is allowed.
ERROR	An error occured in the TES software running on the execution node

  - Active APs
    The number of active Automated Procedures executed on the execution node.
  - Suspended APs
    The number of suspended Automated Procedures loaded on the execution node.

Copyright per DIN 34

- State
  Indicates whether archiving is enabled.
- Close Cycle
  The cycle time in minutes an archive file will be closed automatically.
- Collection

  This entry defines an archive collection.
- Free Disk Space
  Free disk disk space of the execution node's local disk in kilo bytes.

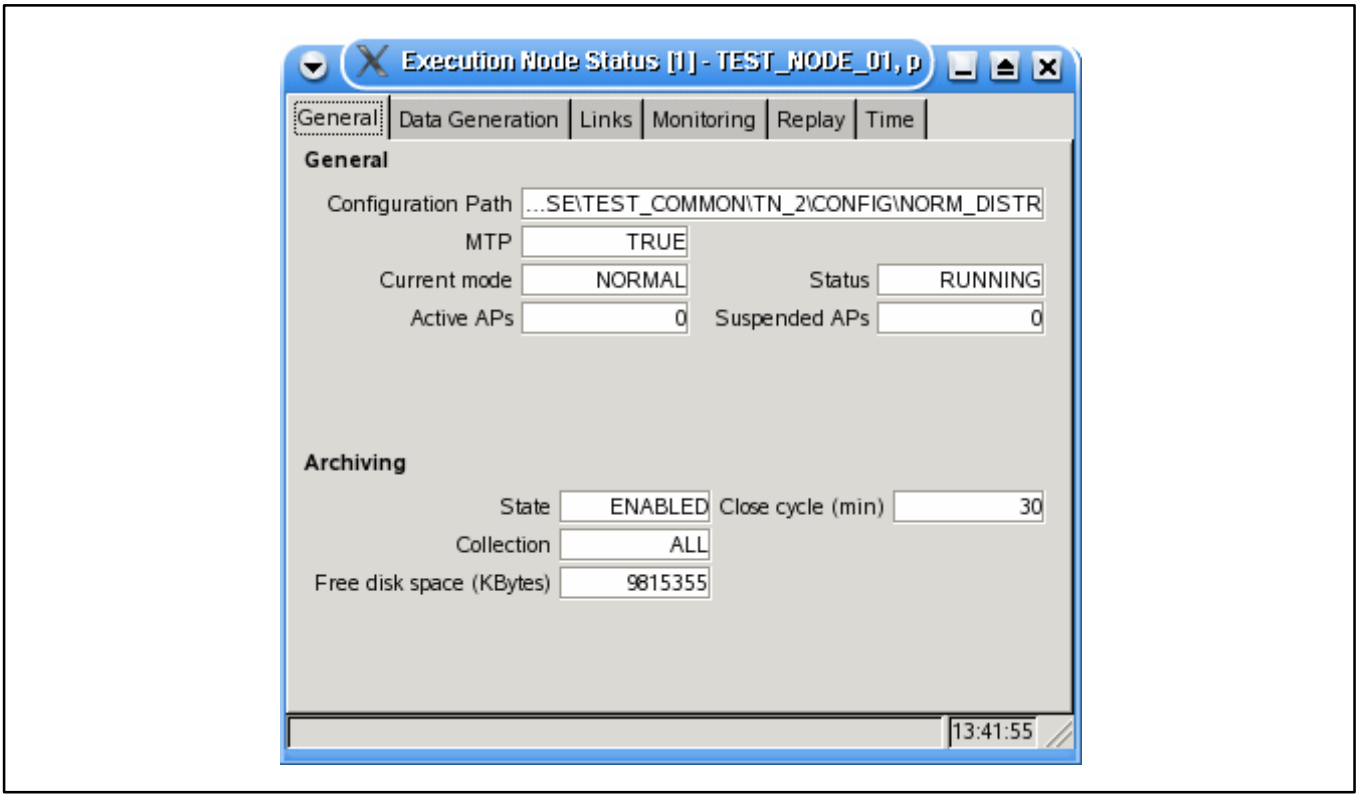


Figure 7–47 : *Execution Node Status – General*

Copyright per DIN 34

- Data Generation
  Displays statistics about Generation Data Units (GDUs) loaded from the data base and the current status (see Figure 7–48).

  LOADED FROM DATABASE

  - Total GDUs
    The total number of GDUs loaded from database.
  - CCSDS TC GDUs
    Number of GDUs with CCSDS (Consultative Committee for Space Data Systems) Tele–Commands loaded from database.
  - Digital GDUs
    Number of digital output GDUs loaded from database.
  - Analog GDUs
    Number of analog output GDUs loaded from database.
  - Bin. pack. GDUs
    Number of binary packet GDUs loaded from database.

  CURRENT STATUS

  - Stimuli sent out
    Number of stimuli sent out since last start (includes erroneous one’s).
  - Stimuli errors
    Number of GDUs with errors in Special Application Software (SAS) since last start.
  - Last error in
    Pathname of the last stimulus that resulted in an error.
  - at SAS
    Name of the Special Application Software reporting the last error.

  VERIFICATION

  - Successful
    Number of GDUs sent with successful verification.
  - Failed
    Number of GDUs sent with failed verification.

Copyright per DIN 34

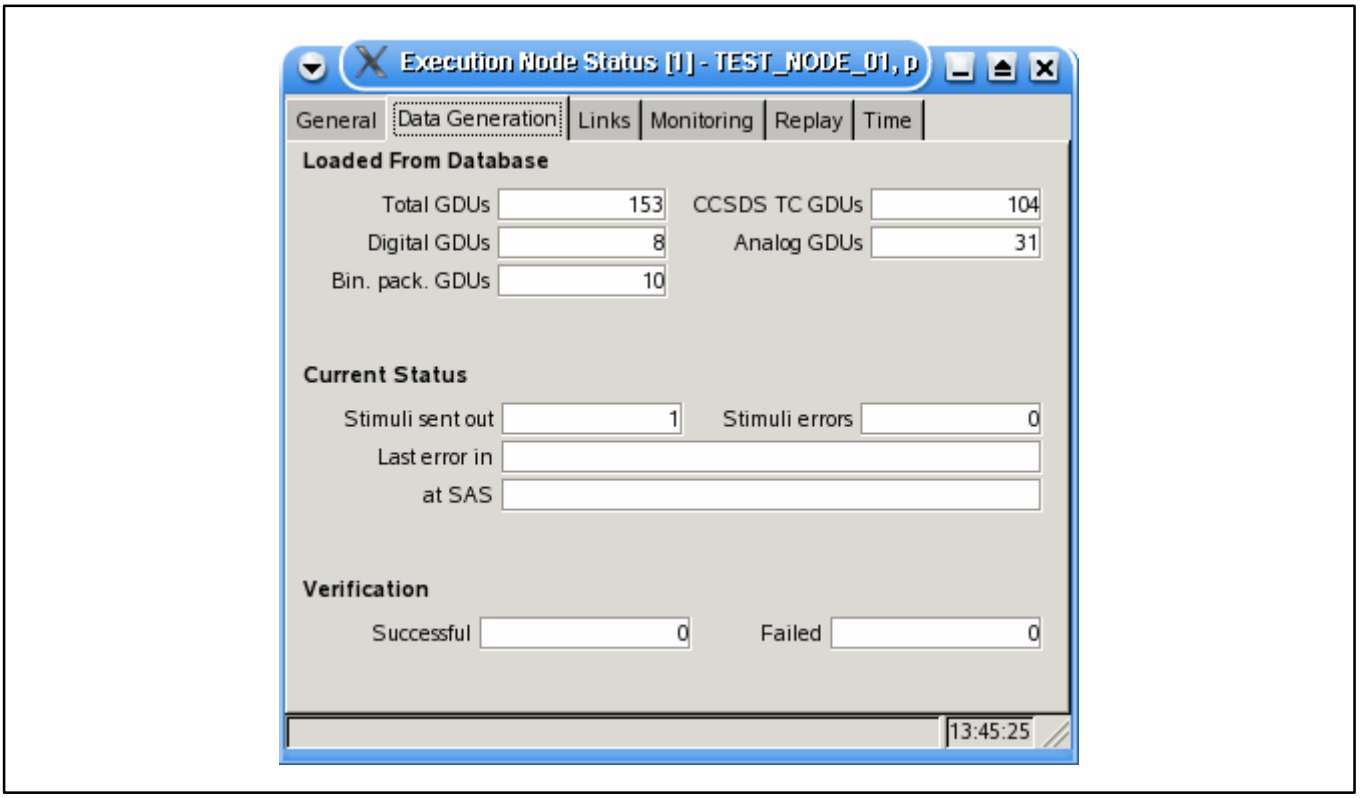


Figure 7–48 *: Execution Node Status – Data Generation*

- Links (see Figure 7–49)
  - Connected workstations
    The number of workstations (Online Control software) connected to the execution node.
  - Connected SAS
    Number of special application software connected to the execution node
  - Connected CIS
    Number of connected CIS applications.

Copyright per DIN 34

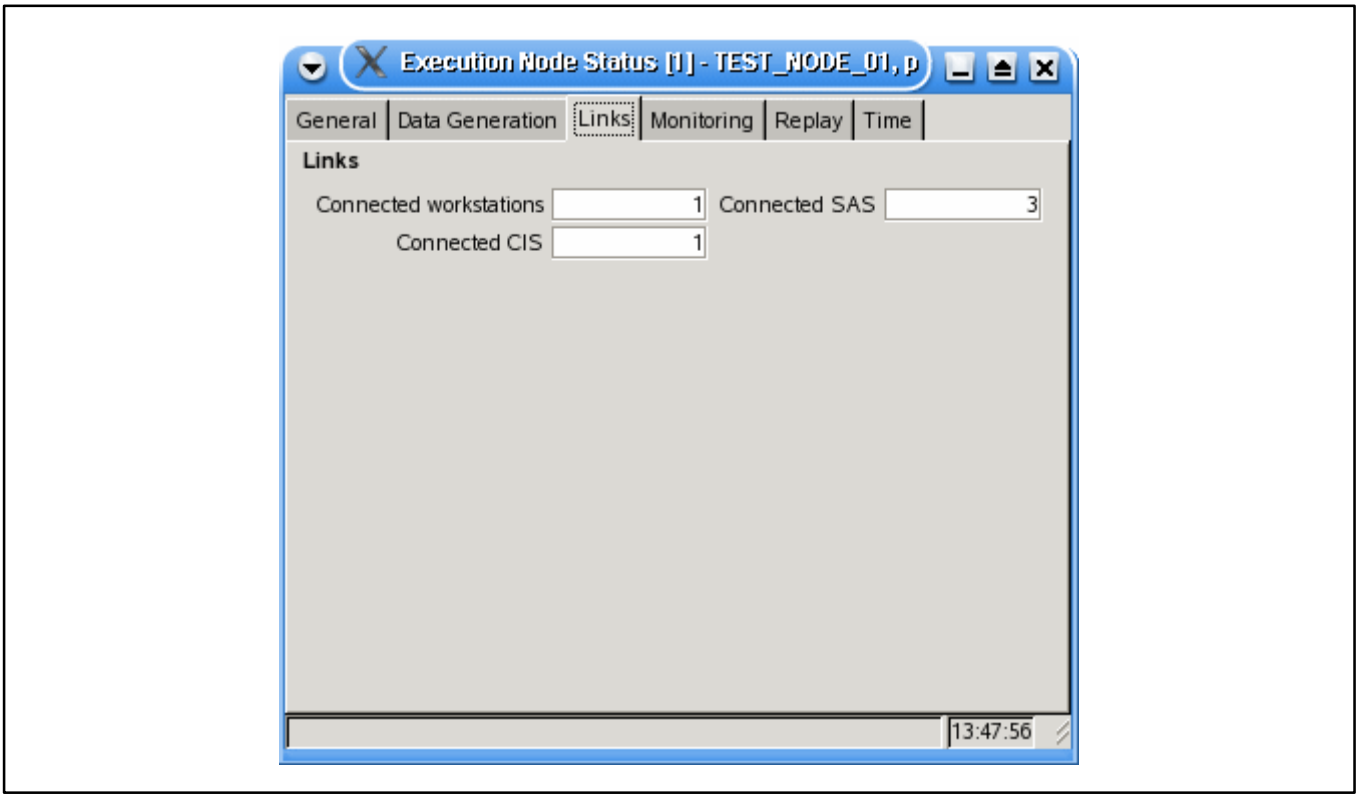


Figure 7–49 : *Execution Node Status – Links*

- Monitoring (see Figure 7–50)

  LOADED FROM DATABASE

  - Monitorable Enditems
    Total number of enditems that can be monitored.
  - Discrete
    Number of discrete enditems that can be monitored.
  - Analog
    Number of analog enditems that can be monitored.
  - Bytestream
    Number of bytestream enditems that can be monitored.
  - Measurements
    Number of measurements loaded from MDB.
  - SW variables
    Number of measurements loaded from MDB.
  - Derived values
    Number of derived value definitions loaded from MDB.

  CURRENT STATUS

Copyright per DIN 34

- Enabled
  The number of enditems enabled for monitoring.
- Not processed
  Number of items with processing set to off.
- Acquired
  Number of measurements currently acquired.
- with EVL
  Number of acquired measurements or software variables with EVL.
- Out of soft limits
  Number of enditems currently out of soft/nominal limit.
- Out of soft limits/since last start
  Number of enditems out of soft/nominal limit since last start.
- Out of danger limits
  Number of enditems currently out of danger/hard limit.
- Out of danger limits/since last start
  Number of enditems out of danger/hard limit since last start.

CONDITIONS

- Defined
  Number of conditions defined.
- Enditems
  Number of enditems carrying conditions.
- Actions triggered
  Number of actions triggered from conditions since last start.

Copyright per DIN 34

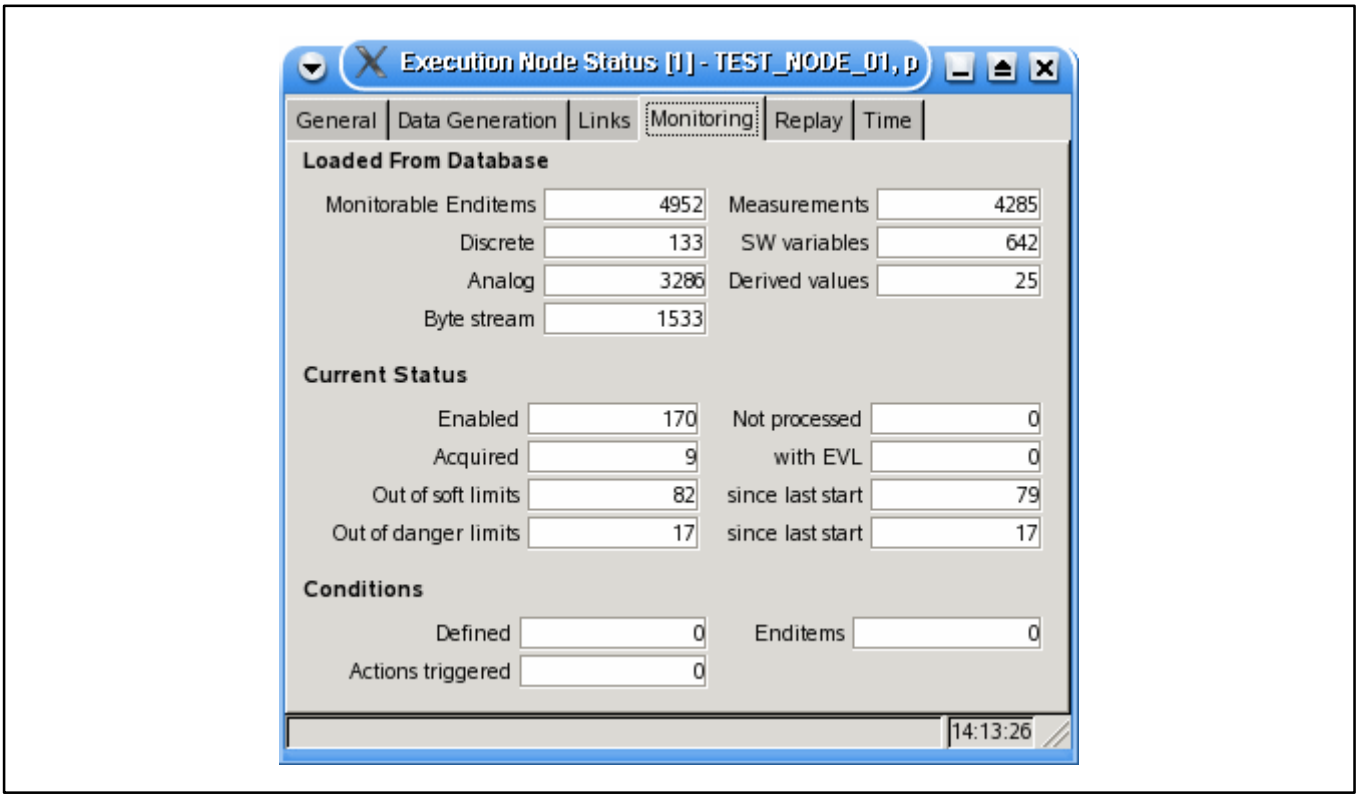


Figure 7–50 : *Execution Node Status – Monitoring*

- Replay Simulation
  - Speed
    Replay speed in percent.
  - Begin time
    Replay begin time selected during initialization.
  - End time
    Replay end time selected during initialization.
- Time
  - Clock synchronized
    Indicates whether the local clock of the execution node is synchronized with the NTP server.
  - SMT status
    Indicates the status of the Simulated Mission Time.
  - External MTU
    Indicates whether the execution node has an external master time unit.
  - System LT synchronized
    Indicates whether local time of the system is synchronized.

Copyright per DIN 34

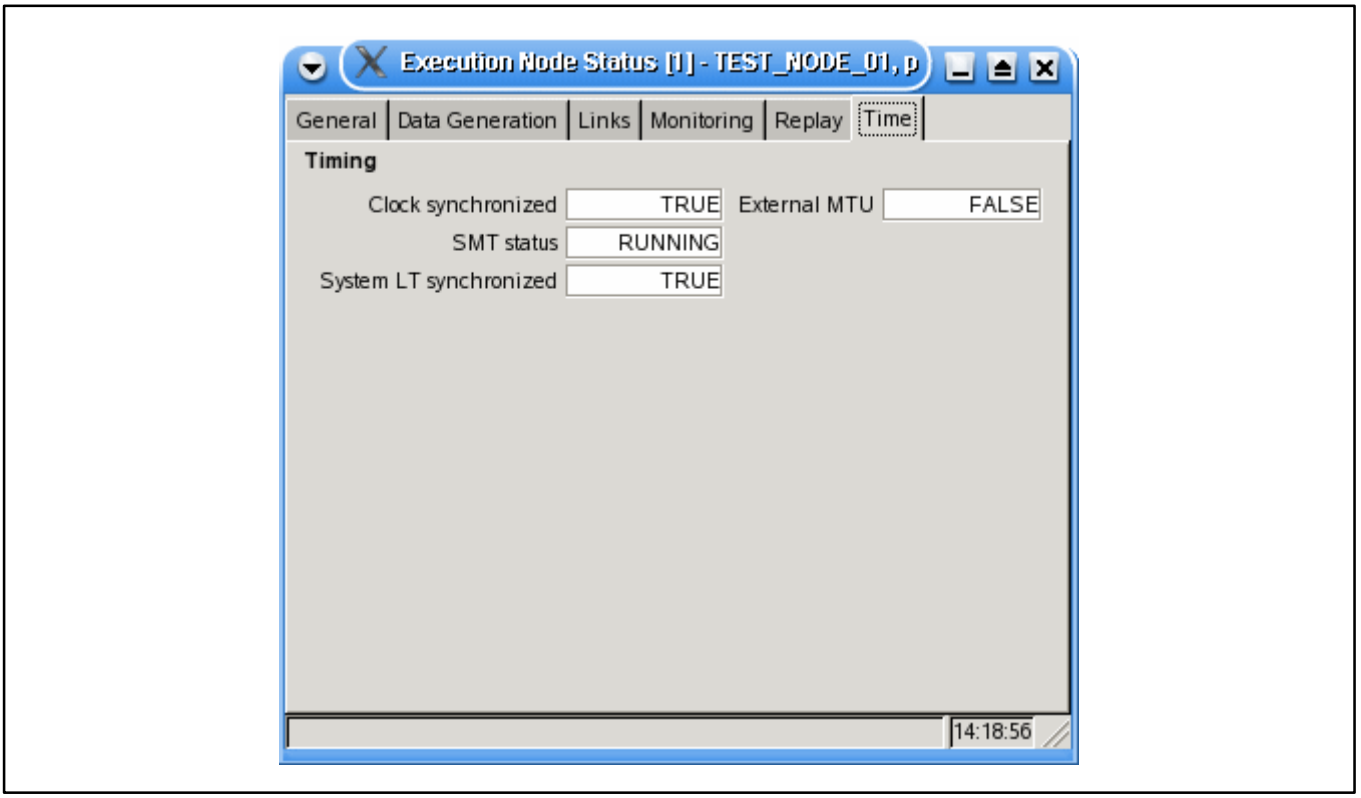


Figure 7–51 : *Execution Node Status – Time*

Copyright per DIN 34

7.3.2.4.13 USS Displays

Unified Synoptic System (USS) Displays are the next generation of synoptic displays integrated with CGS. This section will provide a description how USS is integrated with CGS. For a detailed description of USS itself, see USS User Manual.

7.3.2.4.13.1 Integration with CGS

Since USS does only work on the file system, but not directly on the mission data base, data like USS displays, measurement or command definitions (so called XML SCOE files) must be exported to file system. This is done as far as possible automatically. The file system location where to look up USS displays and XML SCOE files can be defined using USS properties (see USS User Manual, section 1.2 Configuring System Settings). When USS is installed from CGS media, the properties are set as recommended by CGS:

- uss.executor.project.dir=${CGS_HOME}/etc/mda/uss
  The executor's project directory defines the location of the USS displays in file system. When exported from mission data base, the USS displays are stored under a subdirectory called CCU followed by an internal version number. This subdirectory path is automatically added when the USS Executor is connected to CGS via CIS.
- uss.editor.rootfolder=${HOME}/uss–project
  The editor's rootfolder defines the location of the USS displays in file system for editing purposes. The CCU internal version number is automatically added to the rootfolder when the USS Editor is started from IMDB.
- uss.scoe.dir=${HOME}/uss–project/scoe
  The scoe directory defines the location of the XML SCOE files used by the USS Editor. The USS Executor doesn't need XML SCOE files.
- uss.scoe.hkdefinitions=${CGS_HOME}/etc/hkDefinitions.xml
  The scoe housekeeping definitions add all HK definitions of the master execution node to the system interface model.
- uss.executor.hlcl.initsequence=CIS.HLCL.LoginSequence
  The file name of the HLCL init sequence with which to initialize the HLCL interpreter. The path denotes a file that has to be accessible on the node on which the HLCL Interpreter run (usually CIS_01).
  If the given login sequence string is CIS.HLCL.LoginSequence, the login sequence defined in the CGS configuration parameter CIS.HLCL.LoginSequence is used.
  When using a file name as initsequence, it must be set in quotes; e.g. "${HOME}/.cgs/hlcl_login.seq".

The editor's rootfolder should never point to the executor's project directory. When synchronizing displays from data base to file system for execution, the executor's project directory is partly removed to clean up old display files and displays under construction are lost.

Copyright per DIN 34

7.3.2.4.13.2 GWDU to USS Display Conversion

CGS provides conversion of a single GWDU Display as well as a complete CCU to USS display format, but the conversion is restricted to frequently used GWDU display objects and attributes. For a complete list of supported display elements, see USS User Manual, section 7.3.1 GWDU to USS Conversion Details.

- *Display conversion can only be performed with CM Status*

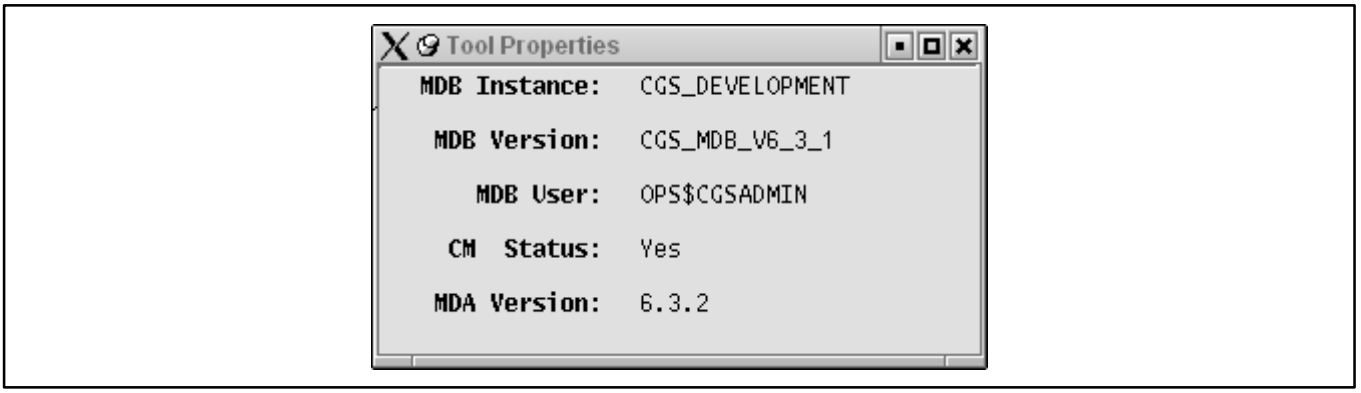


Figure 7–52 *: I_MDB Tool Properties Showing CM Status*

Before migrating to USS Displays it is recommended to make a backup of the mission data base and to create a deep copy of the CCU that can be used to run the conversion process. A copy of the CCU can be created with the CU Version Manager. Remark: quit CU Version Manager before starting the conversion process since it locks the new CCU.

To convert all GWDU displays of a CCU, start I_MDB and open the CCU Versions dialog. Select the CCU and execute

Command –> Tools –> Convert to USS

Copyright per DIN 34

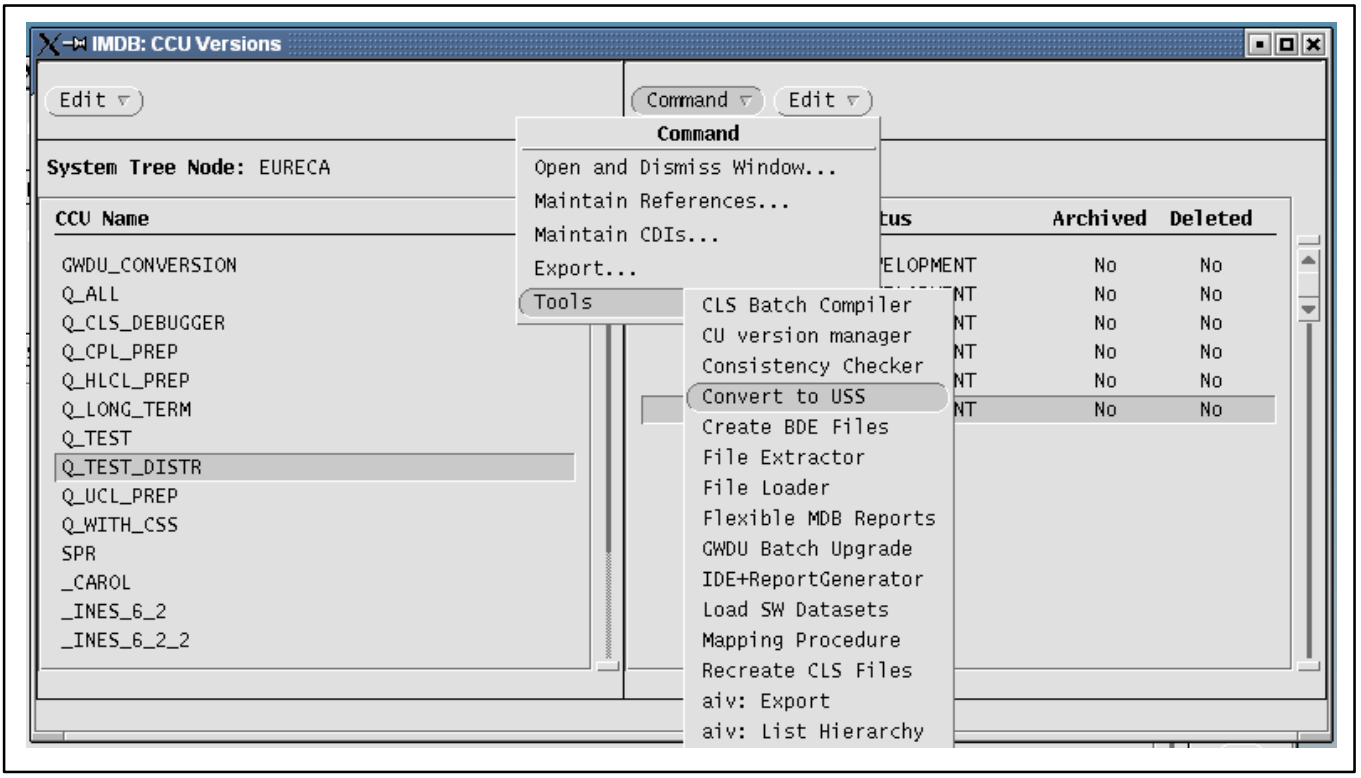


Figure 7–53 *: Convert all GWDU Displays of a CCU to USS*

Using the MDB Upload check boxes on the GWDU to USS Display Conversion dialog, it can be controlled whether the USS display files and/or the USS symbol library shall be uploaded into the mission data base after conversion. When the GWDU displays contain symbols, a corresponding USS symbol library is created with given name. The name must correspond to the mission data base item name format and is automatically converted into upper case letters. When symbol library upload has been selected, the path name is required and an additional check is performed whether the path name is valid and below a CDU and the named symbol library doesn't exist, yet.

MDB Upload Cases

- Upload displays and symbol library
  Used for a complete conversion of GWDU to USS displays.
- Upload symbol library only.
  Used when the GWDU displays shall be converted separately later on.
- Upload displays only.
  Used for separate conversion of GWDU displays.
- Upload neither displays nor symbol library.
  Used for a first dry run to check if the conversion will generate errors or to pack the converted displays/symbol library to take it to a different site.

Copyright per DIN 34

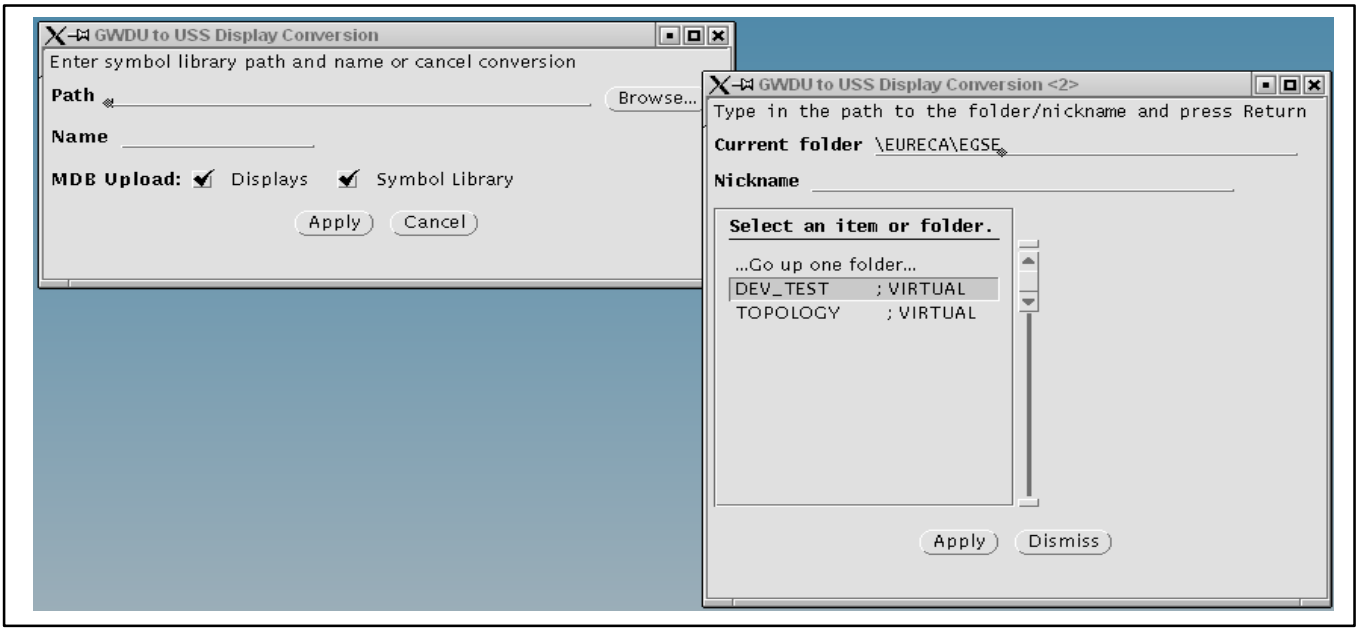


Figure 7–54 *: GWDU to USS Display Conversion Dialog*

After selecting Apply the conversion process continues with XML SCOE file generation. If the XML SCOE files are out–dated or not available (N/A) a confirmation window comes up. If the XML SCOE files are out–dated it may be possible to continue without generation of new files dependent whether all references to commands and measurements are still valid. If no XML SCOE files are avialable it is recommended to create them. Selecting Cancel will not cancel the conversion process, but only generation of the XML SCOE files.

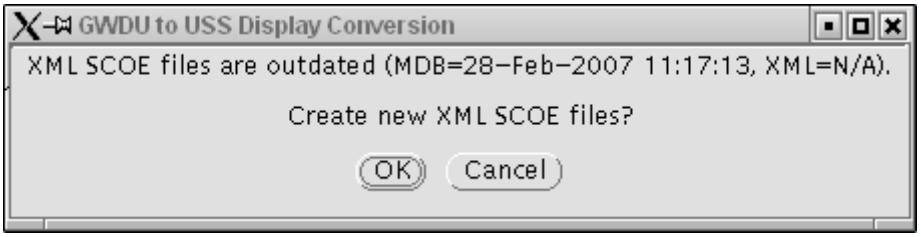

GWDU to USS Display Conversion
XML SCOE files are outdated (MDB=28–Feb–2007 11:17:13, XML=N/A).
Create new XML SCOE files?
OK
Cancel

Figure 7–55 *: XML SCOE Files Generation*

Copyright per DIN 34

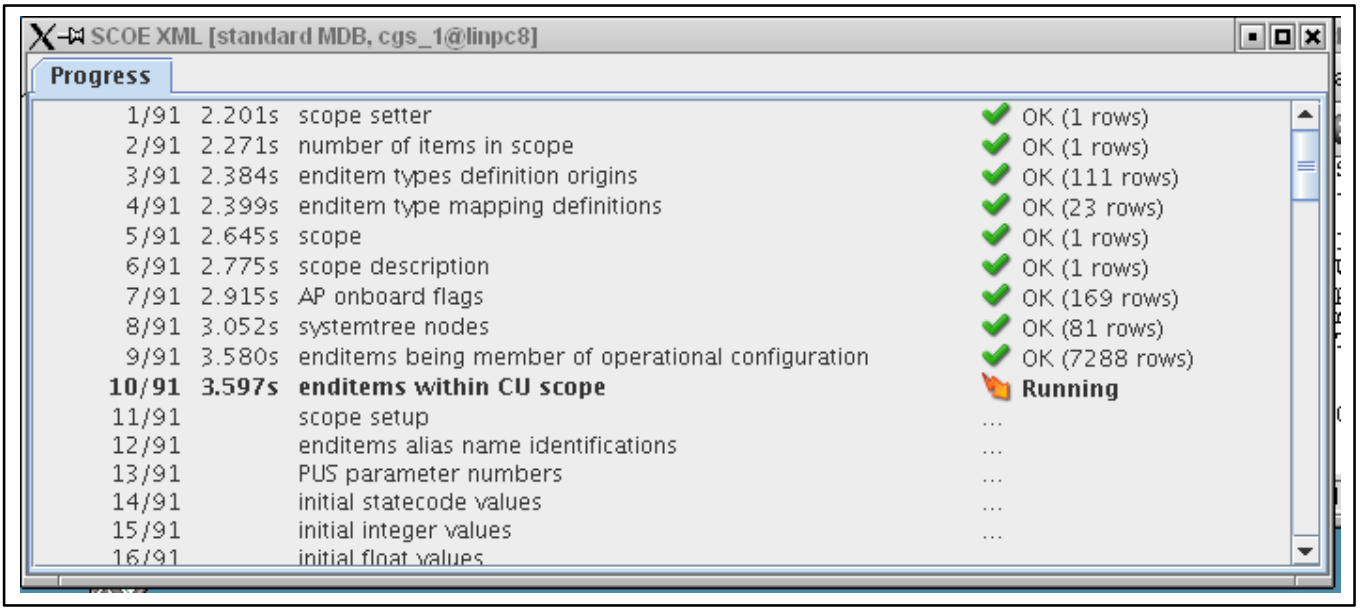

SCOE XML [standard MDB, cgs_1@linpc8]
Progress
1/91 2.201s scope setter OK (1 rows)
2/91 2.271s number of items in scope OK (1 rows)
3/91 2.384s enditem types definition origins OK (111 rows)
4/91 2.399s enditem type mapping definitions OK (23 rows)
5/91 2.645s scope OK (1 rows)
6/91 2.775s scope description OK (1 rows)
7/91 2.915s AP onboard flags OK (169 rows)
8/91 3.052s systemtree nodes OK (81 rows)
9/91 3.580s enditems being member of operational configuration OK (7288 rows)
10/91 3.597s enditems within CU scope Running
11/91 scope setup
12/91 enditems alias name identifications
13/91 PUS parameter numbers
14/91 initial statecode values
15/91 initial integer values
16/91 initial float values

Figure 7–56 *: Generating XML SCOE Files*

The next step is the generation of specific GWDU XML files for each GWDU displays in the CCU scope. These XML files are stored in a temporary directory for the final conversion step. During GWDU XML generation, the GWDU editor pops up for each GWDU display. When finished, it displays a report about generated files.

The GWDU XML files are then converted into USS displays and a corresponding USS symbol library. If a conversion error occurs a window pops up showing how many displays couldn't be converted. At this point it is possible to cancel the conversion process without modifying the mission data base.

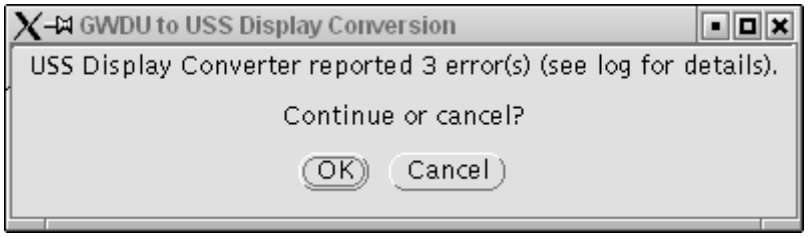

GWDU to USS Display Conversion
USS Display Converter reported 3 error(s) (see log for details).
Continue or cancel?
OK
Cancel

Figure 7–57 *: Conversion Error*

The next step modifies the CCU. It moves all GWDU displays to a virtual node called GWDU_COPIES and removes the alias name.

In the last step the converted USS displays and stored in the CCU under the same path where the old GWDU display was located with the same alias name. Using the path and alias name of the GWDU displays ensures navigation between the new USS displays.

When the conversion process has been finished, a dialog pops up asking whether to remove the directory containing the temporary files. In general, this can be confirmed. In case your GWDU displays include subdrawings (symbols with dynamic behaviour) you must first copy the lib directory from the temporary directory to the USS project directory before deleting the temporary files.

Copyright per DIN 34

Also sophisticated user may use the temporary display files and snapshots for further processing (e.g. to define a new hierarchy for the displays).

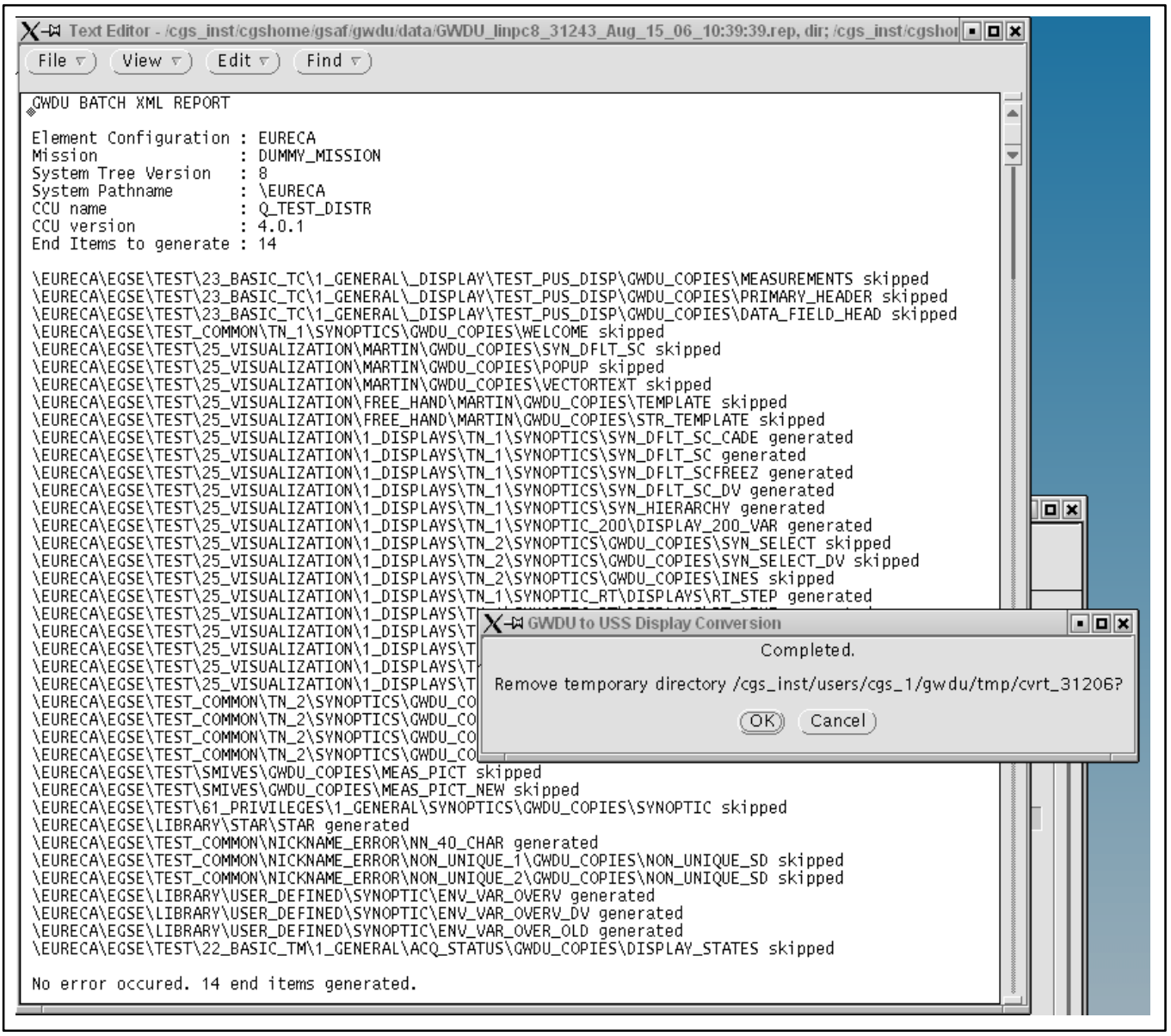


Figure 7–58 *: GWDU Batch XML Report*

Conversion of a single GWDU display can be invoked directly on the end item using the context menu

Tools –> Convert to USS

The conversion steps are the same as already described above, but in this case only display upload should be selected.

Copyright per DIN 34

7.3.2.4.13.3 Starting USS Editor

The USS Editor can be started via either via top level user interface (Task Selector, CGS Start Center) or via I_MDB.

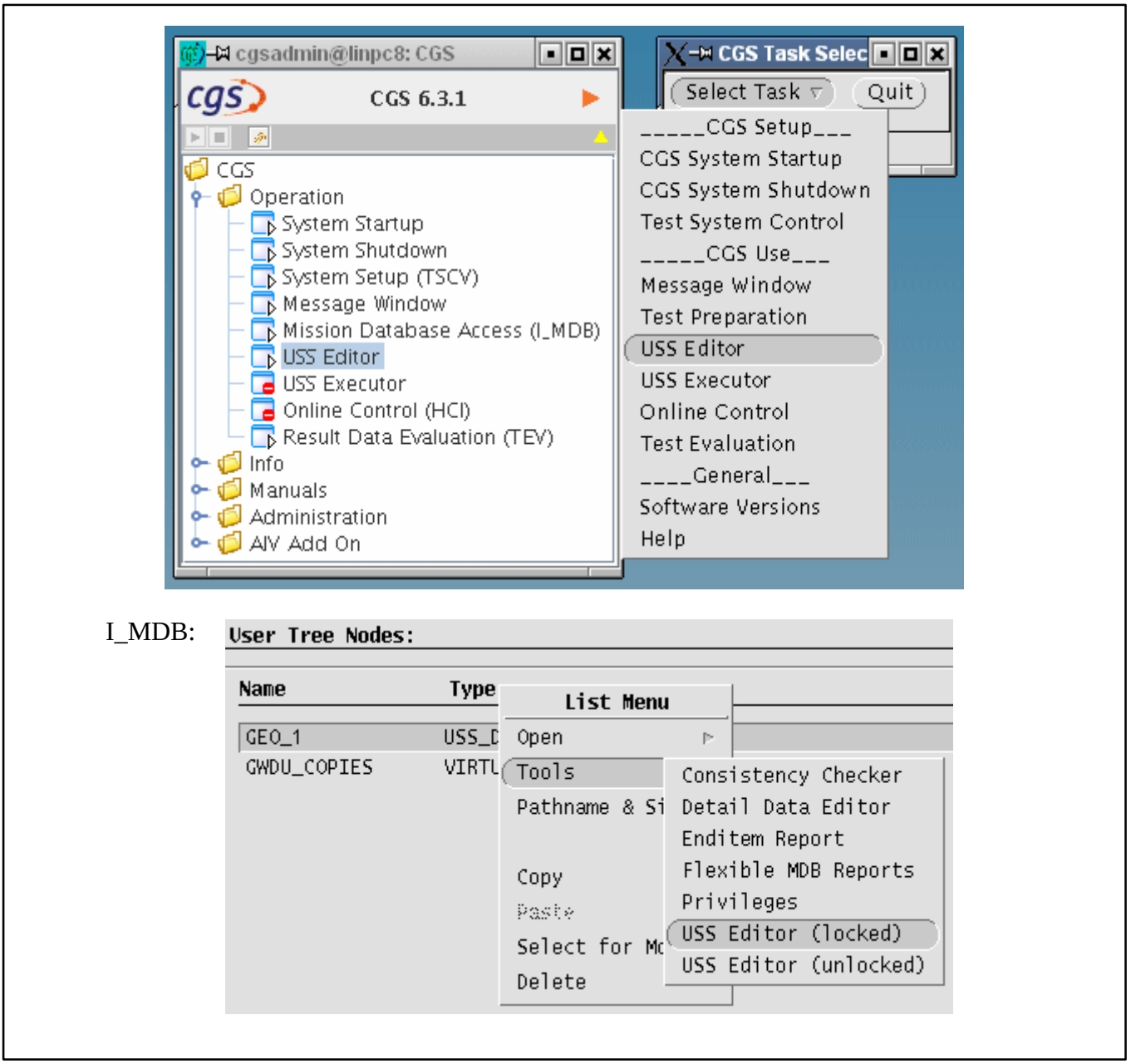


Figure 7–59 *: Start USS Editor*

When starting a USS display via context menu or double–click from I_MDB, XML SCOE files can be automatically generated and selected. When starting USS Editor, it checks whether there is an actual SCOE file and ask for confirmation of the generation process if an outdated or none was found (indicated by date “N/A”).

Copyright per DIN 34

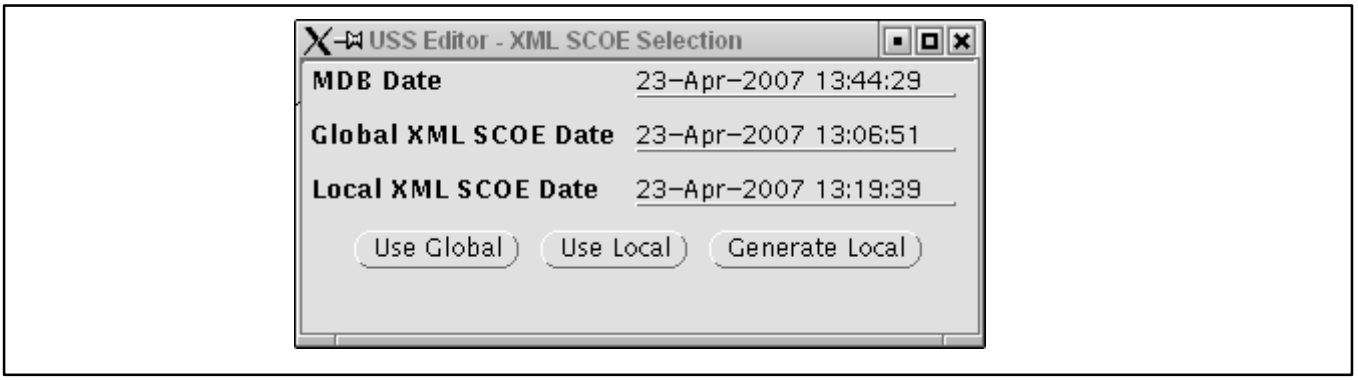


Figure 7–60 *: USS Editor – XML SCOE File Selection*

If the SCOE files are up to date enough (e.g. if only some enditems have been modified not used by the USS Display), either global or local XML SCOE files can be used if available. If recent modifications of the data base must be avialable, select “Generate Local” to create new XML SCOE files. If actual XML SCOE files are avialable the USS Editor is started directly without displaying the XML SCOE Selection dialog. Quitting the dialog has the same effect as selecting “Use Local”. Global XML SCOE files are located at $CGS_HOME/etc/mda/ccu/xml, local XML SCOE files are located as defined by property ”uss.scoe.dir” in file $GSAF_HOME/uss/main/etc/uss.properties.

The dates displayed by XML SCOE Selection dialog are CCU consistency dates. When fresh XML SCOE files have been generated, the CCU consistency date is shown from which they are generated from and not the time of generation.

Copyright per DIN 34

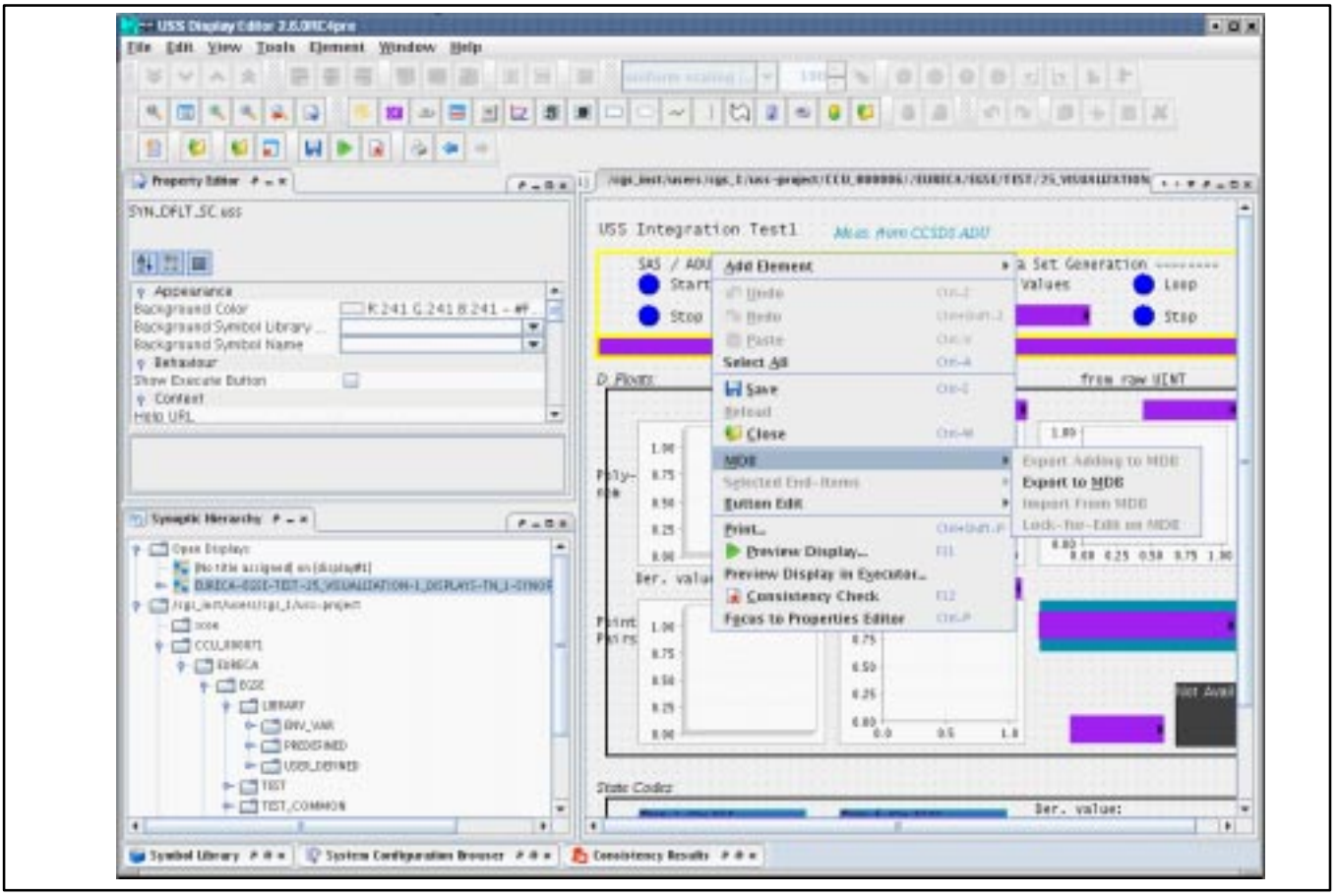


Figure 7–61 *: USS Editor – Export to MDB*

7.3.2.4.13.4 Importing USS Displays from File System into MDB

When a lot of USS displays have been edited in file system, they can be imported into mission data base using IMDB. Import USS Displays can be invoked from the CCU Versions dialog (see Figure 7–62).

There are some prerequisites to be taken into account before importing displays. The root directory from where the import starts must be named according to the CCU internal version number prefixed by “CCU_”, for example CCU_1000215. The directory structure and display/library names must conform to the mission data base naming concept (e.g., all upper case). Since the importer needs to create some temporary files for the library archives, the directory structure must be writable for the user. When the USS Display Importer has been started it shows the CCU internal version. At this point the root directory can still be renamed or created using your favourite file manager according to the required name (see Figure 7–63).

Before beginning the import, the root directory must be selected using the browse button (see Figure 7–64). After selecting the import button, the import process starts and the results are reported in a scrollable text area.

Since importing USS displays overwrites existing ones in data base, it is recommended to work on new CU revisions.

Copyright per DIN 34

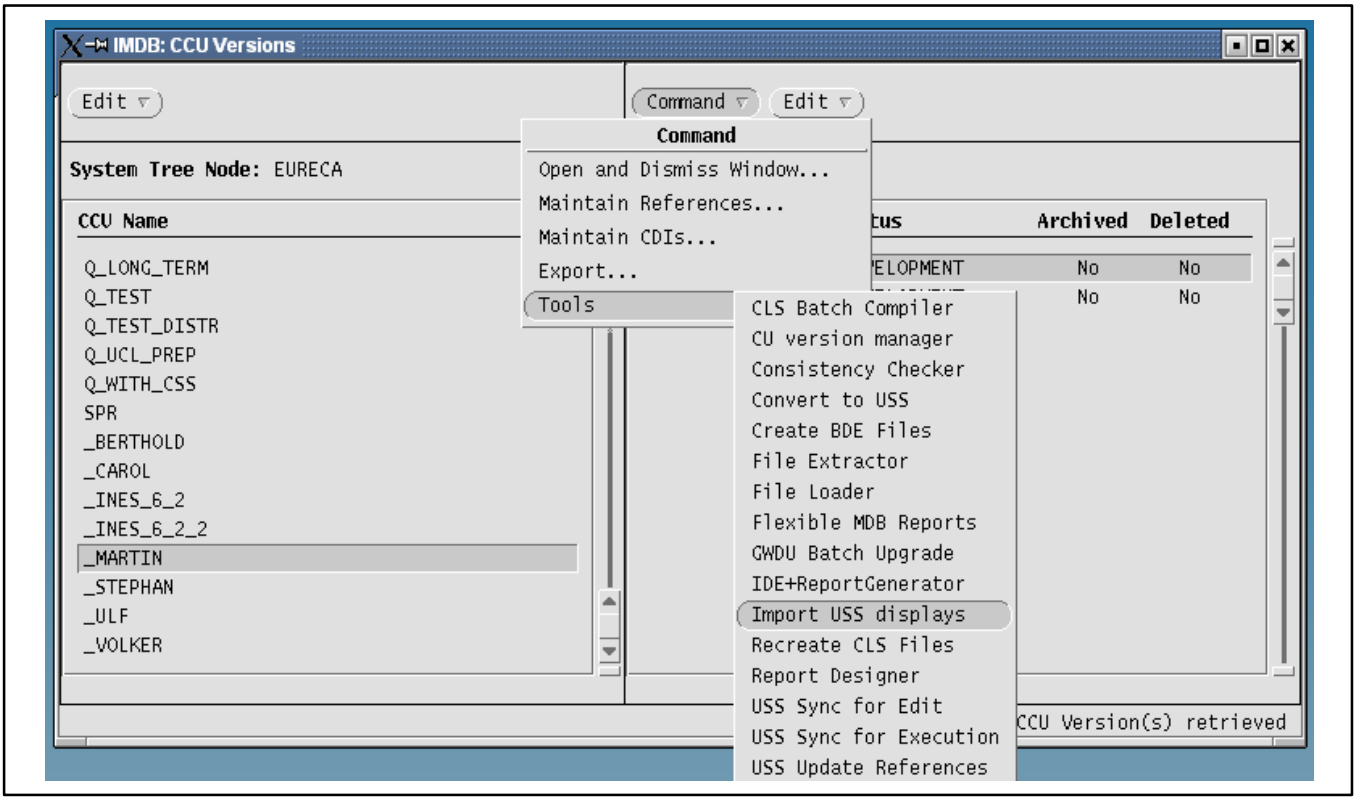


Figure 7–62 : *IMDB – Import USS Displays*

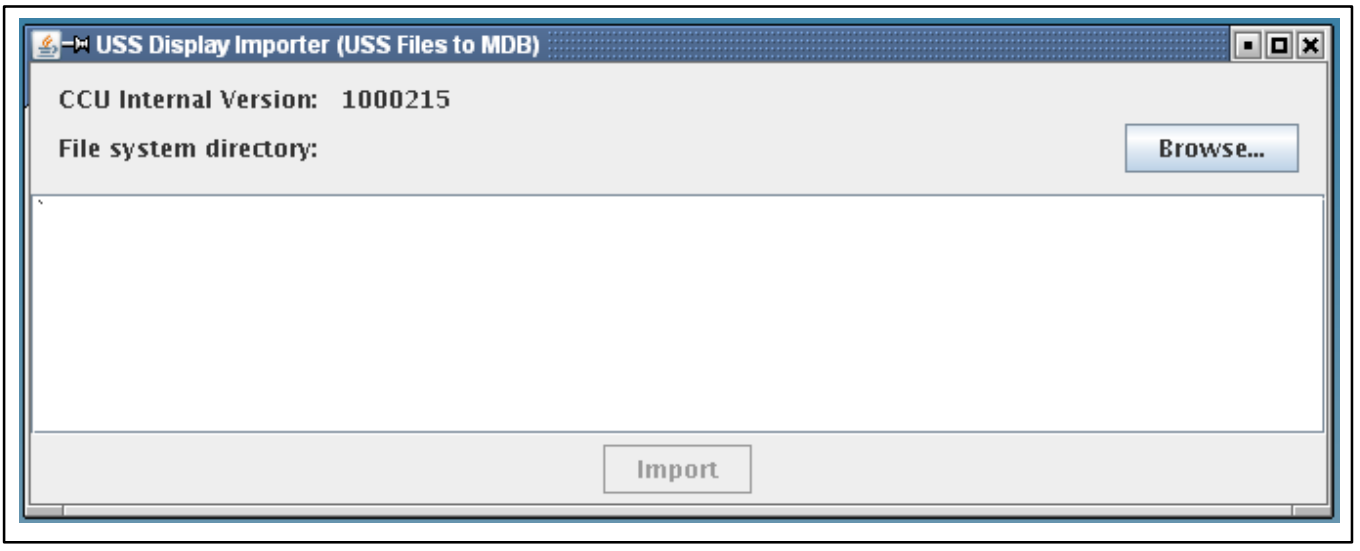


Figure 7–63 : *USS Display Importer*

Copyright per DIN 34

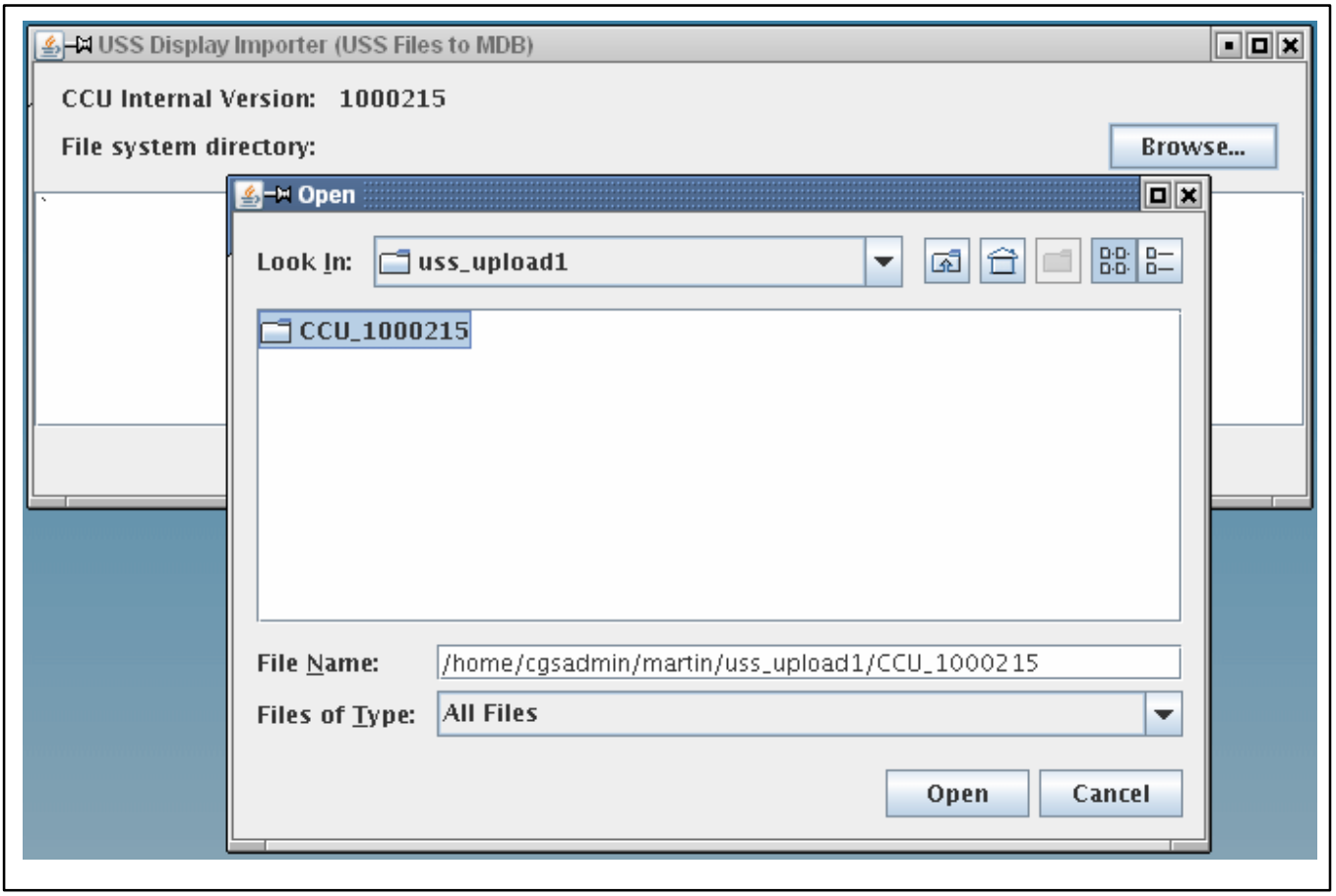


Figure 7–64 : *Browse File System to Select Root Directory*

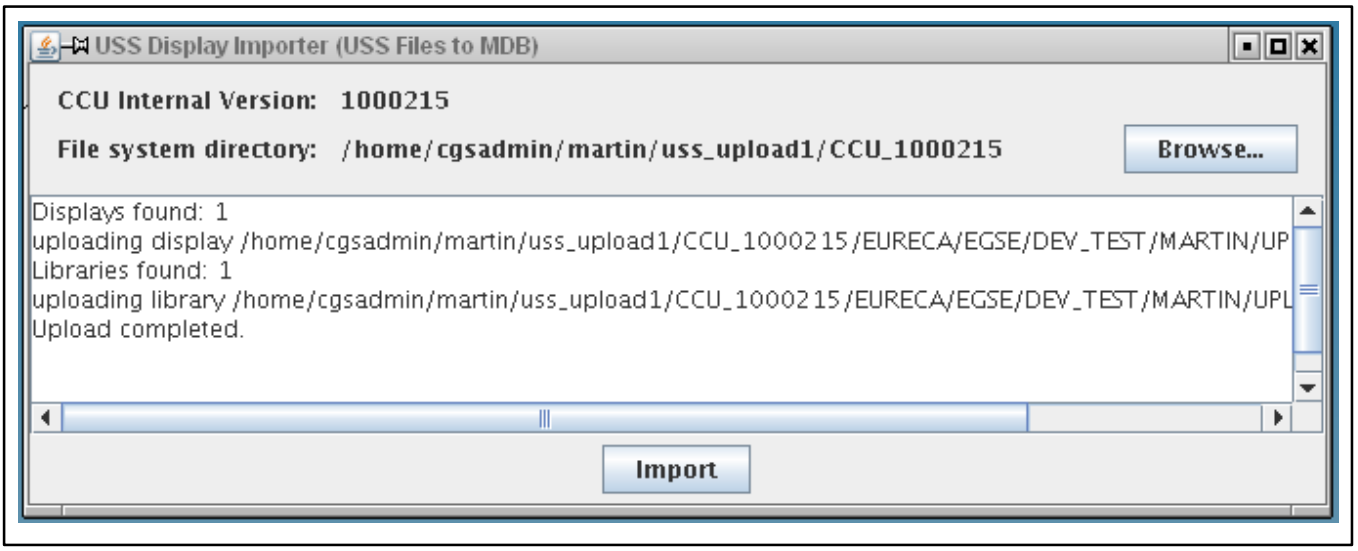


Figure 7–65 : *Importing USS Displays*

Copyright per DIN 34

7.3.2.4.13.5 Updating Parameter References of USS Displays in MDB

If a lot of telemetry parameter have been changed in mission data base, e.g. due to an import from another data base with different SID range, USS Displays can be automatically updated using “USS Update References” tool. The tool can be invoked from IMDB’s CCU Versions dialog like “Import USS displays” (see Figure 7–62).

After starting “USS Update References”, SCOE XML generation is executed if necessary and the USS displays are exported from data base to file system. When the parameter references of the USS displays have been updated in file system, a report is displayed showing the modifications and the USS Displays Importer is started. The reports are stored in user’s home directory prefixed with “uss–ref–”. If the modifications are acceptable, the USS displays can be imported back from file system to mission data base by pressing the “Import” button of the USS Display Importer.

Since importing USS displays overwrites existing ones in data base, it is recommended to work on new CU revisions.

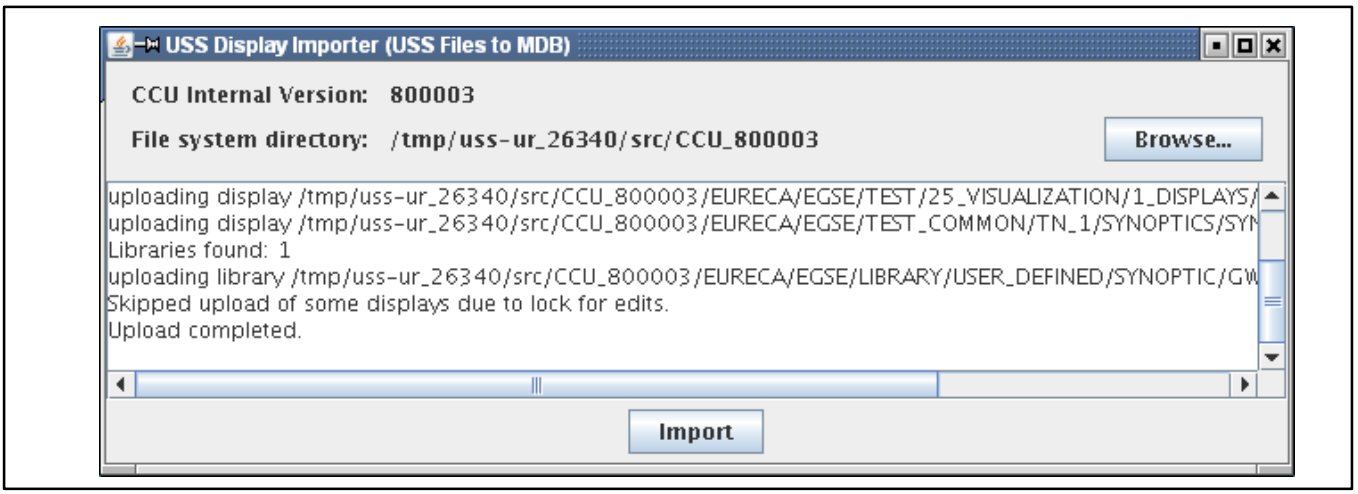


Figure 7–66 *: Importing Updated USS Displays*

If some USS displays are locked for edit by other users, the USS Display Importer reports that some displays have been skipped. In this case quit USS Display Importer, resolve the locks by reverting or submitting the edits, and call “USS Update References” again.

Copyright per DIN 34

7.3.2.4.13.6 Starting USS Executor

The USS Executor can be started via top level user interface like the editor. It is recommended to define an active HCI instance (see section 7.2 Setting–Up the Environment) for each workstation node where to run an USS Executor. If the workstation node (HCI instance) is participating in an active test configuration, the USS Executor will automatically connect to CIS.

When running together with Online Control, USS displays can be displayed/removed via HLCL and UCL commanding using the same commands as for GWDU synoptics. USS displays started through Online Control are included in screen setups, too.

7.3.2.5 Screen Setup Maintenance

To pop up the Screen Setup Maintenance window, select
Application –> Screen Setup

Screen setups are stored as ASCII files within the directory $HOME/.cgs/screen_setup_pool.

The following window applications can be part of a screen setup

a. ADU Status (referred to as ADU_STATUS)
b. AP Status (referred to as AP_STATUS)
c. Clock (referred to as CLOCK)
d. Command Facility (referred to as COMMAND_FACILITY)
e. Database Node Status (referred to as DB_NODE_STATUS)
f. Execution Node Status (referred to as EXECUTION_NODE_STATUS)
g. Explorer (referred to as EXPLORER)
h. Out Of Limit Display (referred to as OUT_OF_LIMIT_DISPLAY)
i. Graph Facility (referred to as GRAPH_FACILITY)
j. Monitoring Window (referred to as MONITORING_WINDOW)
k. Online Control window (referred to as MAIN_MENU)
l. Raw Data Dump (referred to as RAW_DATA_DUMP)
m. System Advisory (referred to as SYSTEM_ADVISORY)
n. SAS Status (referred to as SAS_STATUS)
o. UCL Browser (referred to as UCL_BROWSER)

For each application, one line defining the name of the application, window attributes, and window start parameter is given. The format is based on the command line arguments of XView. Comment lines beginning with a sharp # and empty lines can be inserted.

Screen setups containing a USS Executor are identified by a comment **#USS** in the header of the file.

The following attributes/start parameter are supported:

- –position
  Sets the initial position of the application's base frame in pixels. The upper left corner of the screen is at position (0,0) with the x–axis increasing to the left and the y–axis increasing downward.
  Example: `CLOCK –position 100 200`

Copyright per DIN 34

- –size
  Sets the width and height of the application's base frame. The values are in pixels.
  Example: `AP_STATUS –size 600 150`
- –iconic
  Starts the application in iconic form.
- –node
  Sets the pathname of the default node of Command Facility.
  Sets the pathname of the test node for AP Status, SAS Status, and Execution Node Status.
  Example: EXECUTION_`NODE_STATUS –node \EURECA\EGSE\MTP`
- –default_path
  Sets the default path of Command Facility.
  Example: `COMMAND_FACILITY –default_path \EURECA\EGSE`
- –login_sequence
  Executes the login sequence when starting a command facility. The login sequence can be located in database (\...) or file system (/...) depending on the separator.
  Example for database: COMMAND_FACILITY –LOGIN_SEQUENCE \SEQ\START
  Exampe for file system: COMMAND_FACILITY –LOGIN_SEQUENCE /seq/start
- –measurement_list { list }
  Defines the measurements displayed in the Monitoring Window.
  Example: MONITORING_WINDOW –measurement_list { \MOTOR\CURRENT \MOTOR\TEMPERATURE }
- –file
  Defines the configuration file used by Monitoring Window.
  Example: MONITORING_WINDOW –FILE "$HOME/floats.mon"
- –replay
  Sets the Clock to replay mode.
  Example: CLOCK –REPLAY

The following is an example of a screen setup file:

```
#Version: 7.3.0
#CCU:Q_TEST_DISTR
MONITORING_WINDOW –DESKTOP  1 –FILE "Q25.mon" –POSITION  51  597 –SIZE  449  226
SAS_STATUS –COLUMN_LENGTHS {  108  58  63  230  48  }  –DESKTOP  1 –NODE
\EURECA\EGSE\TOPOLOGY\CONFIGURATION\NODE_LIST\TEST_NODE_01 –POSITION  145  302 –SIZE  526  135
SYSTEM_ADVISORY –DESKTOP  1 –POSITION  147  566 –SIZE  250  158
EXECUTION_NODE_STATUS –DESKTOP  1 –GROUP  5 –NODE
\EURECA\EGSE\TOPOLOGY\CONFIGURATION\NODE_LIST\TEST_NODE_01 –POSITION  150  233 –SIZE  392  327
COMMAND_FACILITY –DEFAULT_PATH \ –DESKTOP  1 –NODE
\EURECA\EGSE\TOPOLOGY\CONFIGURATION\NODE_LIST\TEST_NODE_01 –POSITION  48  115 –SIZE  550  300
MAIN_MENU –DESKTOP  1 –POSITION  0  0 –SIZE  157  78
```

The screen setup maintenance facility allows to maintain such files by saving the actual screen layout to the file. It allows to delete such a file, to open a new screen layout by specifying the name of the setup file or to rename such a file.

Loading a Default Setup

A user specific default setup can be specified by selecting the file chooser button below the Default Screen Setup headline (labelled "(None)" in Figure 7–67). When pressing the button a file chooser dialog pops up and the default setup can be selected by double click or pressing *Open*. The name of the selected default screen setup is then displayed as button label. A default screen setup applies during start up of Online Control. To start it without a default screen setup, press *Clear*. If a screen setup has been specified at command line, the default screen setup is overwritten by that file.

Copyright per DIN 34

Open Screen Setup

To open a screen setup press the *Open* button. This will pop up Open Screen Setup file chooser where a screen setup can be selected. The current windows are all deleted from the screen and replaced by those defined in the selected screen setup.

Since monitoring window configuration files are stored in the same format as screen setups, these files can be loaded, too. Because they are interpreted as screen setup definitions, the behaviour is somewhat different than loading them from monitoring windows: the monitoring window doesn't know the file information and can't display the filename on the header for that reason. When a monitoring window configuration is loaded the current screen is not reset. It is started in addition to existing windows.

If a window manager is used that supports virtual desktops like KDE, the desktop number of a window can be saved in screen setups. This feature is controlled by the virtual desktops check box of the screen setup maintenance window. If the virtual desktops box is checked when saving a screen setup, the desktop number of each window is included in the screen setup definition file. If such a file is loaded again, the windows are displayed in the desktop they were assigned to. If the virtual desktops box isn't checked when loading a screen setup file, all windows are assigned to the current desktop.

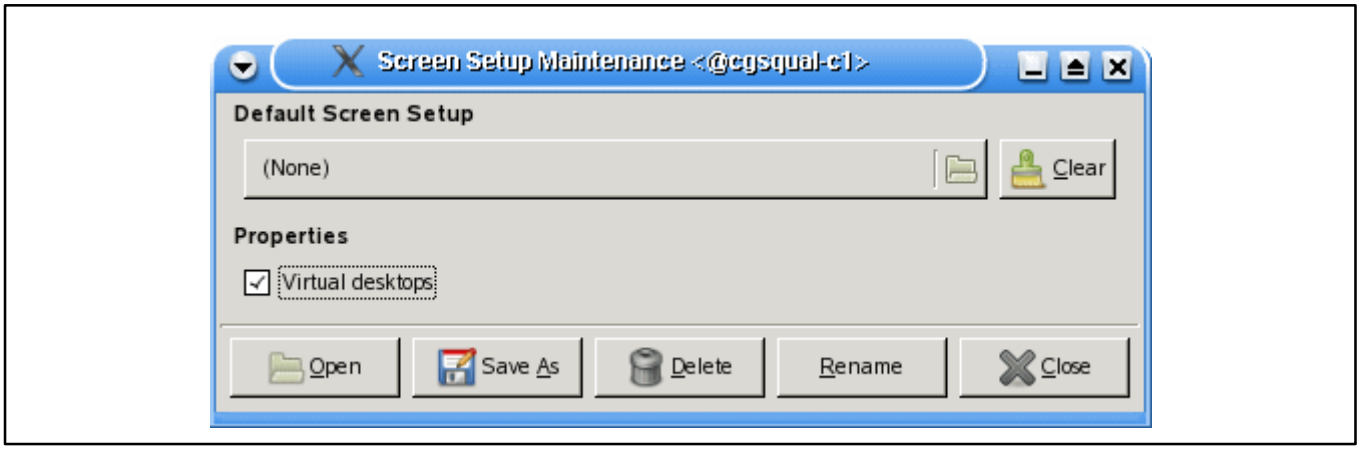


Figure 7–67 *: Screen Setup Maintenance*

If the screen setup was saved in a different configuration, a CCU Conflict dialog is opened (see Figure 7–68). When open is confirmed, the screen setup is loaded but some applications may not start or report errors if database references can't be resolved. To suppress the configuration check for a specific screen setup file, e.g. when using a basic setup containing no configuration dependant information, edit the screen setup file (e.g. using the command "vi $HCI_HOME/data/screen_setup_pool/basic") and remove the line containing the CCU identification (e.g. #CCU: MY_NAME).

In general, it is not recommended to load screen setups from different configuration. Some applications may not be started because database references can't be solved.

Copyright per DIN 34

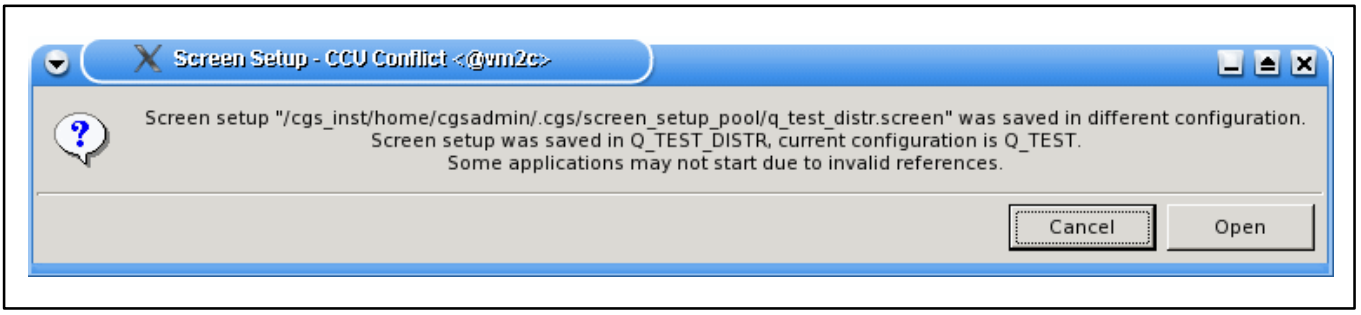

Screen Setup - CCU Conflict <@vm2c>
Screen setup "/cgs_inst/home/cgsadmin/.cgs/screen_setup_pool/q_test_distr.screen" was saved in different configuration.
Screen setup was saved in Q_TEST_DISTR, current configuration is Q_TEST.
Some applications may not start due to invalid references.
Cancel
Open

Figure 7–68 : *Screen Setup – CCU Conflict*

Delete Screen Setup

To delete a screen setup press the *Delete* button. This will bring up a Delete Screen Setup file chooser. After selection of a file and pressing Delete it is deleted from the setup pool after confirmation.

Save Screen Setup

To save the actual screen layout press the *Save As* button. This will pop up a Save Screen Setup file chooser on which the name of the new screen setup can be entered or an existing one selected. The names are case sensitive, ”demo” and ”Demo” are not the same. After selection of the *Save* button on the Save Screen Setup file chooser the current Online Control applications are scanned and the new screen setup definition file is stored in the setup pool. The saved screen setup may be called up later again using the Open operation or by selecting the name as default screen setup.

Rename Screen Setup

To rename a screen setup select the corresponding item from the Select Screen Setup to Rename file chooser and press the *Rename...* button. You can then enter on the Rename Screen Setup dialog the new name of the screen setup. After selection of the *Rename* button of the dialog the new name will apply. and the selected screen setup is renamed in the setup pool.

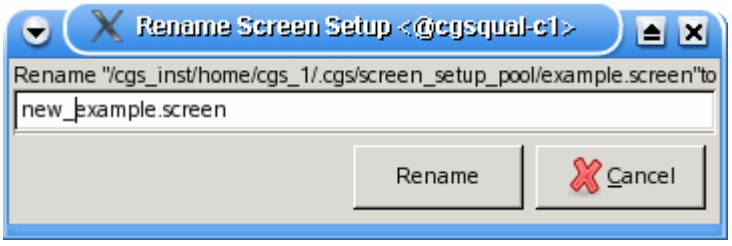


Figure 7–69 : *Rename Screen Setup Dialog*

Loading Screen Setup from Command Line

To start Online Control with a given screen setup, use the –screen_setup parameter. E.g.:

$HCI_HOME/bin/common/start_hci –screen_setup $HOME/25_RT_Displays

7.3.2.6 Starting Applications via HLCL/UCL

All window applications mentioned in the Screen Setup Maintenance section can be started by HLCL/UCL commands, too. To start a window application via START_HCI_APPLICATION (HLCL) or START_PROGRAM (HLCL/UCL), it must be prefixed by “HCI.”, e.g.

Copyright per DIN 34

```
START_HCI_APPLICATION "HCI.RAW_DATA_DUMP","-POSITION 200 100", FALSE
START_PROGRAM PROGRAM:"HCI.CLOCK"
```

The window attributes/start parameter are the same as used by screen setups. To easily get the start parameter for a certain window application, the window can be online configured and stored in a screen setup. The corresponding line in the screen setup can then be used as start parameter for the HLCL/UCL commands prefixed by "HCI.". If the screen setup line contains quotes '"', they must be duplicated (escaped) to correspond to HLCL/UCL syntax.

7.3.2.7 UCL Browser

The UCL browser interface allows the update of UCL related MDB content during a session and the reloading of the changed content on an execution node.

The browser can display derived values, automated procedures and HLCL sequences. The UCL browser–>editor allowes re–editing of these entities. The actual generated I–CODE after re–editing can be reloaded to the related execution node.

The UCL Browser Tool can be selected as:

Application –> UCL Browser

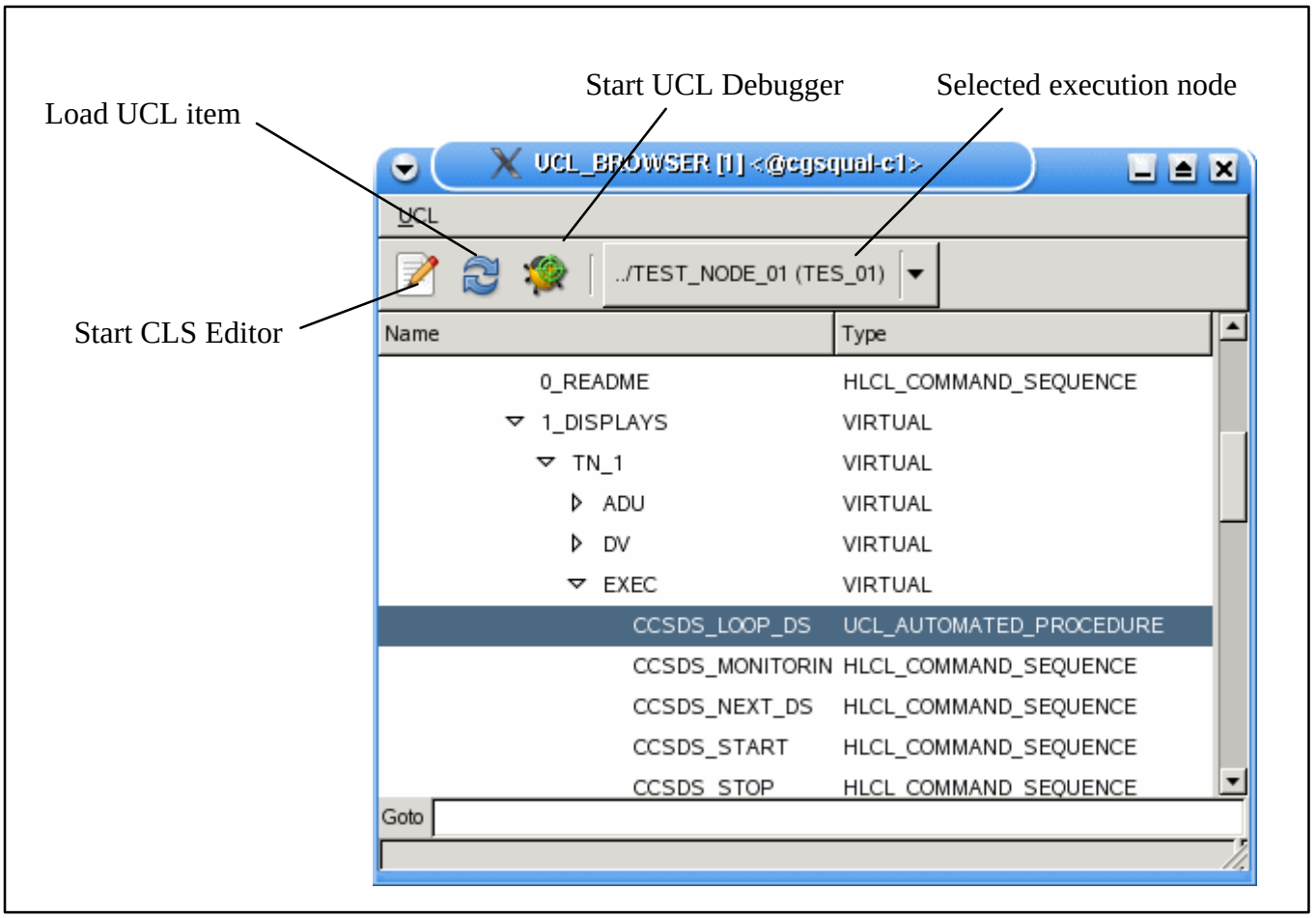


Figure 7–70 *: UCL Browser Window Example*

The UCL Browser functions are available via the UCL menu or by selecting the corresponding toolbar buttons.

Copyright per DIN 34

The “Start CLS Editor” and “Start UCL Debugger” buttons start the corresponding tools for the selected item.

The *Load UCL item* button causes the selected execution node to reload the actual I–Code of the selected item.

The Goto text input field can be used to quickly select an item in the browse area. When the text matches a pathname, nickname, or short identifier the item is displayed and selected in the browse area.

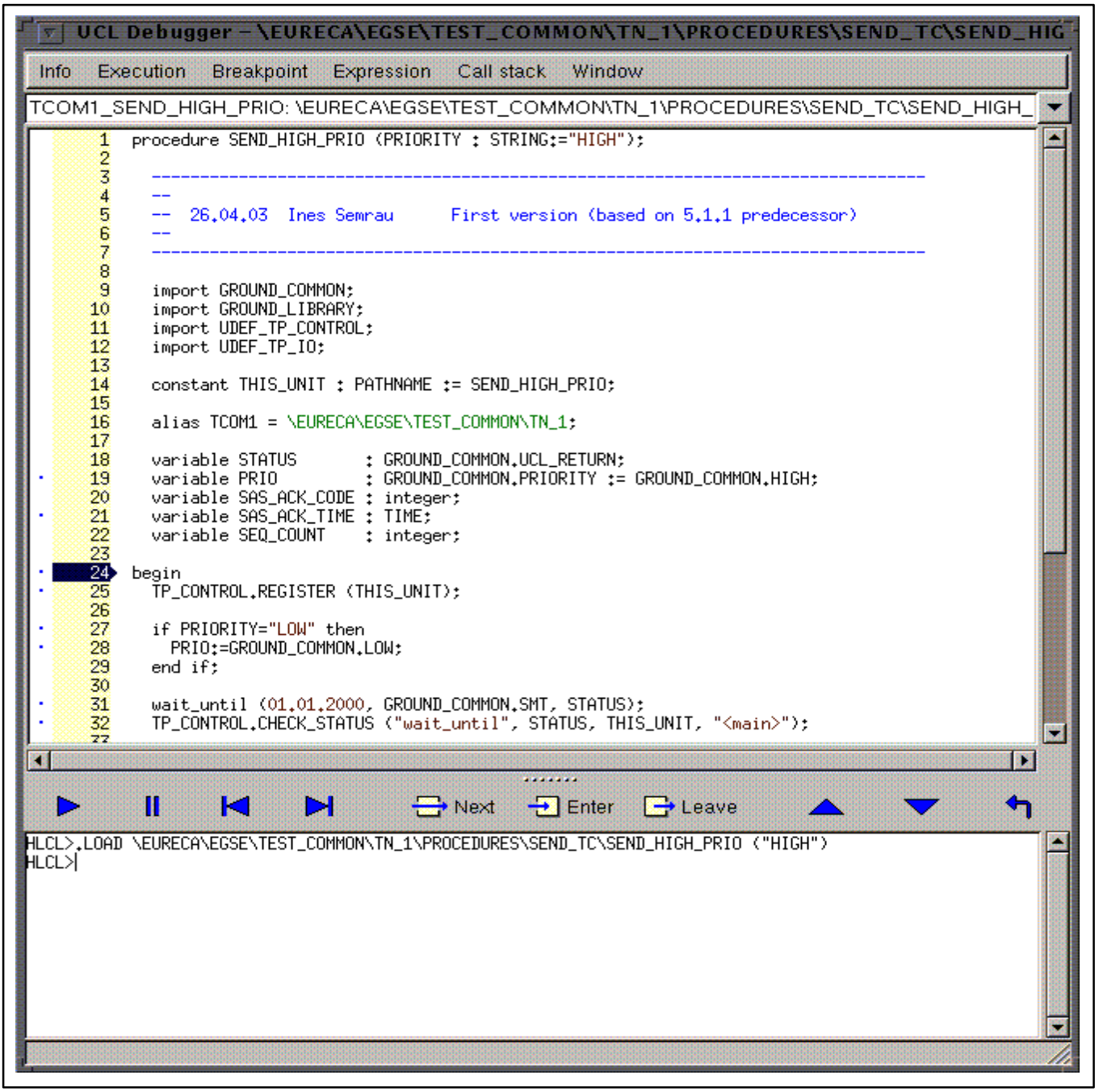


Figure 7–71 : *UCL Debugger*

The UCL visual debugger provides support for debugging UCL–based automated procedures.

Copyright per DIN 34

The debugger provides tooltip help functions. The debugger provides code high lighing and standard debugging code navigation and viewing options for UCL procedures and libraries used. The debugger is described in a dedicated user manual (see chapter 2.2). If the selected UCL procedure requires parameter input, a dialog window will request data input from the user.

7.3.2.8 Quit

To quit online control, select

Application –> Quit

This will pop up a confirmation window as shown in Figure 7–72. Exiting Online Control will terminate all its applications and services that have been started, but it will not terminate online help manual viewers or user services.

Figure 7–72 *: Quitt Online Control*

7.3.2.9 Input Dialog

The Input Dialog (see Figure 7–73) is invoked by the UCL library procedures READ_MESSAGE_FROM_USER and READ_NUMBER_FROM_USER or for critical command authorization.

The Input Dialog can be used to enter data to a running AP. The Input Dialog displays a prompt and as tooltip what type of data is requested by that AP (in this case a string value). When the data was entered select either *Ok* to commit the input or *Cancel* to cancel the input.

When used for telecommand authorization, it prompts for password input. The characters typed in are masked by *.

Typing <Return> key after data input has same meaning as selecting the Ok button.

Copyright per DIN 34

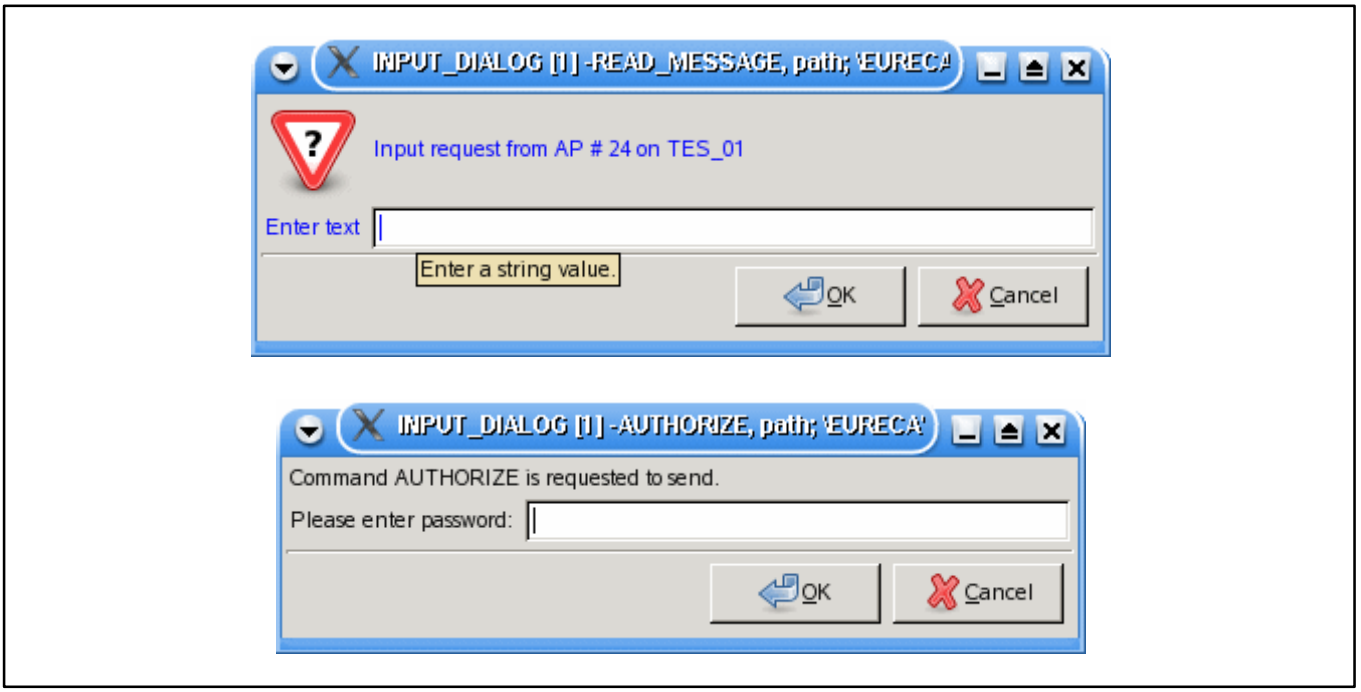


Figure 7–73 : *Input Dialog*

Sample for read message from user (as used for Figure 7–73):

```
        variable AP_STATUS : UCL_RETURN;
        variable MESSAGE   : STRING(255);
              ...
READ_MESSAGE_FROM_USER
      (PROMPT          : ”Enter text”,
      WORKSTATION      : \\,
      USER_ENTRY       : MESSAGE,
      OPTIONS          : “–position 100 200 –foreground #0000FF –image
question“,
      STATUS           : AP_STATUS);
```

Legal options for Input Dialog are

–position x y
–size width height
–foreground color

Color can be specified by name (Unix command showrgb provides a list of names) or by RGB values (e.g. #0000ff).

–image name

Name can be info, warning, error, question, or any Gtk Stock item name like gtk–execute, gtk–help, gtk–network,...

Copyright per DIN 34

7.3.2.10 Telemetry Window Footer Configuration

For *telemetry related windows* (e.g. Out of Limit Display, Monitoring Window, HLCL Commanding), the window footers are configurable to show application specific status information (e.g. indication of playback or realtime mode).

The file $CGS_HOME/etc/cgs_configuration.xml needs to be modified to define application specific status information.

This section will provide a small example to add datasource status information to OTC window footers. The first step is to register the new status value as house keeping (HK) value in the execution software. This is done by adding (unhide and modify) the corresponding parameter in $CGS_HOME/etc/cgs_configuration.xml:

```
Houskeeping.Names                    = USER_DEF_HK_1
Houskeeping.USER_DEF_HK_1            = 1150
Houskeeping.USER_DEF_HK_1            = STATE_CODE
Houskeeping.USER_DEF_HK_1            = Realtime
```

`Houskeeping.USER_DEF_HK_1` is a list element with three entries for housekeeping Id = 1150, housekeeping type = STATE_CODE and housekeeping value = Realtime. The list parameters are of type String.

```
Houskeeping.StatusDataFormat         = Datasource: %1150%
```

(see example in cgs_configuration.xml for group Housekeeping too)

The StatusDataFormat may contain any text format including positional references to the StatusDataList's HK names. The implementation of this example will result in a window footer as shown in Figure 7–74.

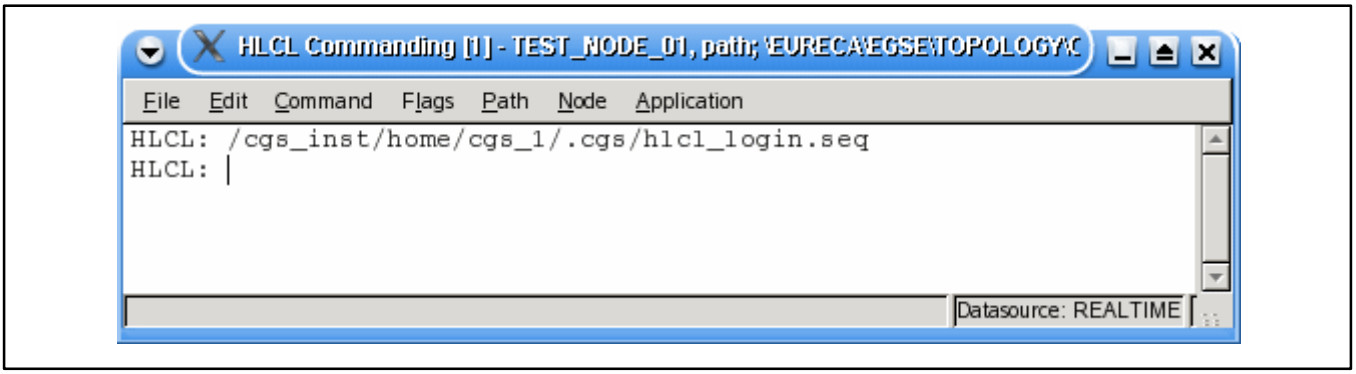


Figure 7–74 *: User Defined Window Footer*

Copyright per DIN 34

7.3.3 Test Execution: Monitoring, Archiving and AP Execution

7.3.3.1 General

During online execution of test, the software running on test nodes (TES) provides automatic data processing and monitoring according to definitions loaded from the MDB.

Test nodes are configured by the TSCV software tool (see dedicated chapter 7.2).

Automated Procedures may be executed by TES and SAS may be controlled by TES that run in parallel to TES on the same node or on remote nodes.

In the following the functions provided by test nodes are described w.r.t. to the interface to the user and to understand the system behaviour.

The test execution activities can be performed in three different modes of operation:

- Normal mode
- Simulation Mode
- Replay Mode

For description of the modes refer to chapter 7.1.2 (on page 7–1).

All modes support the monitoring and data processing function, the execution of UCL commands and the visualisation of data. Sending of GDU to SAS is only performed in NORMAL mode (while in other modes the corresponding operations are accepted, but simulated).

7.3.3.2 Monitoring and Data Processing

After having the test node initialised (i.e. having setup the test configuration and the test nodes switched to an operational mode), TES is ready to **acquire** data from SAS. The **SAS** must be setup into operational state using UCL commands. Then the data may be acquired from SAS. Each single item (resp. the whole name tree, a subtree or a monitoring list) may be activated by the START_ACQUISITION command.

TES will then **request** the data from the SAS, supplying the header information and the physical address given in the ADU_DESCRIPTIONs in the MDB. This includes the ADU_ID, the CCSDS Primary Header and the CCSDS Secondary Header information. In case the data is received from onboard links, the SAS will either request the data from this link, using the given information as address information, or just route the data to TES. In case the data is received from frontend devices , most probably the physical address information will be used. ADUs are then sent cyclicly or asynchronously to TES, where the ADUs are processed to fetch the values from the data part.

Data may be received after **interruption**. In the next ADU after the interruption the SAS may signal a problem to TES, which sets the interrupted status then to all end items referring to this ADU.

The SAS may further indicate an outage problem on ADUs to CGS (via a specific procedure in the TES_API). TES then sets the acquisition status of all end items of this ADU to "**static**".

End items may be enabled for **processing** or not. Processing means fetching of raw value, calibration, monitoring and delivering data for display. Processing may be controlled via HLCL/UCL commands or automatically via conditions.

TES will **calibrate** the data as defined for each end item in the MDB. Calibration means conversion from raw values into engineering values applying a calibration curve. Curves are specified either as polynoms or

Copyright per DIN 34

as point pair sets. For Point Pair Sets , calibration is done by linear interpolation between the point pairs.
Raw values are fetched from the data part of *Unstructured* ADUs, from the whole *CCSDS packet* (CCSDS/PUS) or from the raw value list transferred in *Structured* ADUs. Raw values are interpreted acc. to the raw value types and bit_size information specified for the measurements.

For CCSDS packet, the system can be configured to verify the checksum of the incoming CCSDS telemetry by setting to true the configuration parameter TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM in the *configuration.xml* prior starting the system. In that case, if the checksum is found to be incorrect (for those CCSDS packets having a checksum), a message will be generated. Another configuration parameter allows prevent or not the extraction of data from CCSDS packets having an incorrect checksum. This parameter is named *TES.KERNEL.DATA_PROCESSOR.ADU.PROCESS_ON_INCORRECT_CHECKSUM* in the *configuration.xml*. Of course this configuration parameter only applies in the case where the system is configured to check the checksum.

TES will **archive** the raw data packets (ADU) into archive files, if archiving is enabled. Archive Files are transfered to the central DB Server node as specified when archiving was enabled (default: every 30 minutes)

When received, the ADUs are queued to ensure their processing in the correct order (of reception) and to be overcome a peak ADU rate. The length of the queue is set by default to 500 ADUs. It can be modified by changing the configuration parameter *TES.KERNEL.ADU.QUEUE.MAX_NUMBER_OF_QUEUED_ADUS* prior starting the system.

Further end item values may be received from SAS or APs which write to **software variables**. TES itself may calculate its own values from HK values or generate derived values (see below).

TES will **dispatch** the engineering data to workstations which show them in synoptics. Only those values are transferred which are requested by the loaded synoptics. Together with the engineering value, the acquisition status, processing status and monitoring status is transferred

Monitoring may be activated via the ENABLE_MONITORING command. TES then will monitor each new value according to the active limit sets defined. Exceptions will be generated, if limits are violated.
If Danger Limits are violated, a message of type ”ALRM” is generated. If soft limits (monitor limits) are violated, a message of type MSG is generated for each single violation. If actions are defined for exceptions, they will be executed: APs will be started, GDUs or GDU lists will be sent or user defined messages will be sent to the message handler. This actions will performed with the highest priority.

The AP to start from a monitoring exception should not have any parameter, otherwise it will be rejected. The GDU to generate must as well be either without parameter or have default parameters (that are then used). This is due to the fact that the action is executed automatically and that the user has not the possibility to define the parameters.

For the GDUs and GDU list, as in case they are generated interactively, a timeout can be given, a default timeout is used in case they are generated as action from a monitoring exception. This default timeout value can be modified in the *configuration.xml* under the name TES.KERNEL.GDU.HANDLER.ISSUE_TIMEOUT_WHEN_MONITORING_EXCEPTION prior starting the system.

Each monitor limit will have a message allocated which is to be generated by the monitor in case of limit violation (Exception). An exception message is to be handled always as a message to the HCI and to the DBS logging service . A standard exception message is to be generated by TES. This message contains the actual value of the end item, the limit violated, the applied limit set and a user specified message.
The message is sent via HCI to the message services, and will display in message_handler.

Copyright per DIN 34

It is possible to define delta limits and a counter indicating allowed violations of the soft limit set before an exception is generated. A hard (danger) limit violation is always reported.

Range limit checking and delta limit checking are performed in parallel, i.e when monitoring a measurement, the result can be one of the following states:

- nominal
- out of danger limit
- out of danger delta
- out of soft limit
- out of soft delta
- out of danger limit and out of danger delta
- out of danger limit and out of soft delta
- out of soft limit and out of danger delta
- out of soft limit and out of soft delta

The **Alarm Count** specified in the MDB for the end item is used only for soft(nominal) limit checking. It specifies the number of consecutive out of limit situations before an exception is generated.

Copyright per DIN 34

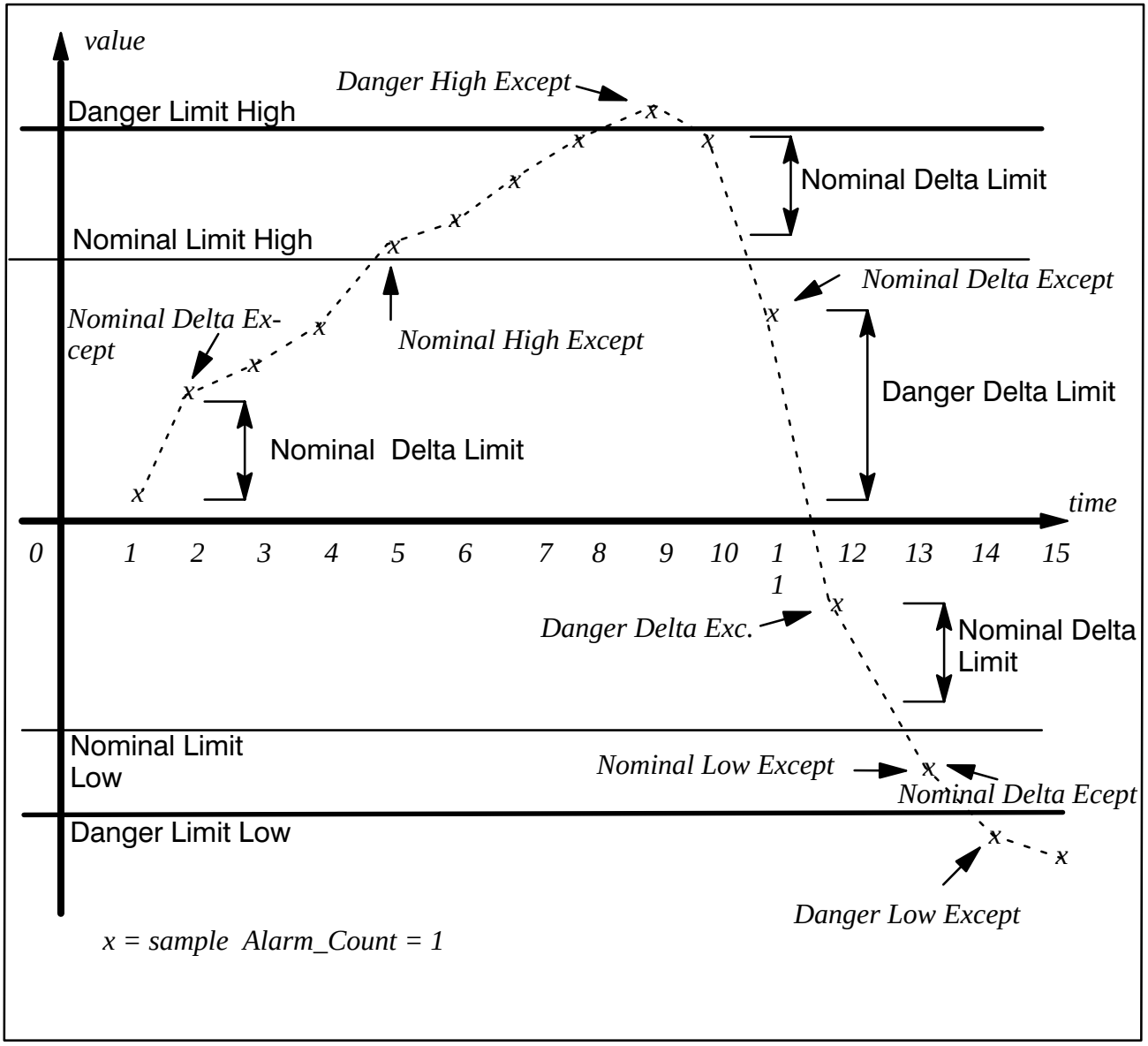


Figure 7–75 : *Concept of End Item Monitoring*

General Rules for message/action generation of **analog** values:

Whenever a delta limit is violated (and was not violated in the previous sample and no interruption of data acquisition occurred), a message/action is generated. When a danger delta limit is violated, no nominal delta message/action is generated.

Whenever the value crosses the limit border (from in_limit to out_of_limit) between two samples, a corresponding message/action is generated. If both the nominal and danger limit borders are crossed, only one message/action shall be generated (for danger limits). For nominal limits, the msg/action shall be generated after ALARM_COUNTER samples are still in that same out of limit state.

Copyright per DIN 34

After a danger message/action is generated, the next danger message/action can only be generated in one of the following cases:

- The measurement goes from above danger high limit to below danger low limit.
- The measurement goes from below danger low limit to above danger high limit.
- The measurement leaves for at least one sample the out of danger limit area (into out of nominal limit or into nominal range) before going back into the out of danger limit area.

All in limit transitions (transition from 'out of limit' to 'in limit') can be logged depend on *configuration.xml* parameter *TES.KERNEL.LOG*... (see section 7.3.3.10 on page 7–149).

After the generation of a message/action for a danger high limit violation, the generation of message/action from out of nominal high limit is inhibited until the measurement goes at least for one sample below the nominal high limit.

After the generation of a message/action for a danger low limit violation, the generation of message/action from out of nominal low limit is inhibited until the measurement goes at least for one sample above the nominal low limit.

After the generation of a message/action for a nominal high limit violation, the generation of a new message/action from out of nominal high limit is inhibited until the measurement goes at least for one sample below the nominal high limit.

After the generation of a message/action for a nominal low limit violation, the generation of message/action from out of nominal low limit is inhibited until the measurement goes at least for one sample above the nominal low limit.

After a danger delta message/action is generated, the next danger delta message/action is generated only if in between the delta monitoring of the measurement indicates either a nominal status or an out of soft delta status.

After a danger delta message/action is generated, the generation of a nominal delta message/action is inhibited until the delta monitoring of the measurement indicates a nominal status.

After a nominal delta message/action is generated, the generation of a nominal delta message/action is inhibited until the delta monitoring of the measurement indicates a nominal status.

Rule for **discrete** values:

Whenever the value is not as the expected value, and was as the expected value in the sample before ALARM_COUNTER samples, but was different for all the previous ALARM_COUNTER samples, a message/action is generated.

SW Variables

SW Variables are end items that are generated within software. The source of the value is an AP or SAS writing to it or an internal HK value that is written by TES. The values are updated whenever a request is received from AP or SAS. SW Variables can be monitored in the same way as measurements.

Copyright per DIN 34

HK values

The TES will provide the update service for each of its housekeeping variables.

HK values will be referenced to the user (HCI, AP or SAS) via SW variable: each HK value has an internal identifier. This identifier is given in the definition of a SW variable. TES will then provide the HK value whenever a user refers to the SW variable. Through this, it is also possible to monitor the value of housekeeping variables (e.g. free disk space) and to take appropriate actions in case the value goes out of limit.

Some HK values are modified on change (the one that are directly under the control of the test node software). Other HK values are modified cylcicly. The update cycles of those HK values is defined in the *configuration.xml* and their values can be modified inside that file prior starting the system. The configurable parameters of that file are in group *TES.KERNEL.HK_VALUE_PROVIDER* for values *UPDATE_CLOCK_PERIOD* for the HK values related to the time, *UPDATE_DISK_PERIOD* for the HK values related to the local disk space, *UPDATE_DBS_PERIOD* for the HK values related to the status of the test result database and the database server, *UPDATE_TSS_PERIOD* for the HK values related to the SMT and its status and *TES.KERNEL.UCLI_CONTROLLER.HK_VALUE_UPDATE_PERIOD* for the UCL related HK values.

Note, especially for the last parameter that a high frequency may lead to additional CPU load of the test node.

Derived Values

Derived Values can be defined in the MDB. Standard UCL is used to define the expressions and the dependencies to other end items. Expressions are converted to I–Code and executed as any other UCL code. Some restrictions apply to the expressions (see ch. 6.6.5.2.3 on page 6–231).
The expression is calculated whenever a new value is received (via ADU or as a result of write operation to software variables) or (if a cycle is defined) if the update time is reached.

Derived Values may refer to other Derived Values.

A derived value that is referencing several measurement from one ADU will only be calculated once when the ADU is processed because derived values referencing measurements are calculated at the end of the processing of the ADU once all measurements have been processed.

States of end items

An end item can have the following states:

Acquisition

VALID	the item is valid
NOT_ACQ	the item is not acquired
NOT_RECVD	the item hasn't been received yet after start of acquisition
NOT_MAINTAINED	the item is not known on the test node
INVALID,	the value of the item is invalid
STATIC	there is no update to the item at the moment

Processing

ENABLED	Processing (Calibration) is enabled
DISABLED	Processing (Calibration) is disabled, either due to a specific command or due to a condition

Copyright per DIN 34

Monitoring

ENABLED	Limit Checking is enabled
DISABLED	Limit Checking is disabled

Derived Values inherit their acquisition status of a the values they are dependent on: E.g.: Whenever one of the those values becomes INVALID, the depend derived value is also INVALID. In addition, an error during calculation of the derived value may also lead to the INVALID status.

In case the **interruption** flag is set for an ADU, or the ADU has been indicated as **static**, the resp. end items shall not be subject to delta monitoring until two consecutive items without interruption flags are received.

In case the **processing is disabled,** the **monitoring status** is not affected: In case monitoring is enabled, limit checking is applied for every new valid engineering value: If processing is disabled, however, no new engineering values are generated.

End item Grouping

End items (measurements, sw variables and derived values) will be addressable as a set (group). This will be implemented by taking the name hierarchy in the DB and the monitoring lists:

- o An enable/disable command with incomplete name–tree information (i.e. specifying a name that is not on the leaf level) will be interpreted in such a way that all parameter below that name are to be enabled/disabled.
- o An enable/disable command giving a monitoring list (in UCL: 'monitoring table') as a command parameter will be interpreted to enable/disable all end items given in the list.
- o Both methods may be used in parallel, thus allowing specification of all end items under a specified subtree which belong to a specified monitoring list.

Grouping can be applied to many operations on measurements, sw variables and derived values.

Conditions

Conditions may be specified that allow to enable/disable processing of end items, defining the applied limit sets of end items or start APs. Conditions are true or false when a specified expression becomes true or false. An expression is a simple expression of an end item with a comparator and a value (e.g. "\end item\a >= 0"). Conditions are defined for an end item whose value triggers the action. Several Conditions may be active in parallel for the same end item. Actions may refer to single end items or groups (virtual trees, monitor lists).

Whenever a condition is triggered, a message is produced in the Event Logging, not in the message window. In case the system is heavily using conditions and the messages are not desired, their generation can be disabled by modifying the parameter *TES.KERNEL.DATA_PROCESSOR.MESSAGE_ON_CONDITION* in the *configuration.xml* prior starting the system.

Online Modification

The following modifications are possible online during test execution without regeneration of Test Node (SCOE) files:

- It is possible to enable/disable the limit checking for a parameter by an online HLCL/ UCL command.
  Together with the enable/disable command, the limit set to be applied may be given that overwrites the set given as 'default set' in the measurement definition.
- New limit values may be specified online via UCL.

Copyright per DIN 34

- The acquisition address for an ADU can be modified with respect to
  - SAS Name
  - APID in CCSDS Primary Header
- Conditions can be generated or modified

The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup)

7.3.3.3 Sending of Generation_Data_Units (GDU)

In the follow flow chart the GDU processing is figured in overview.

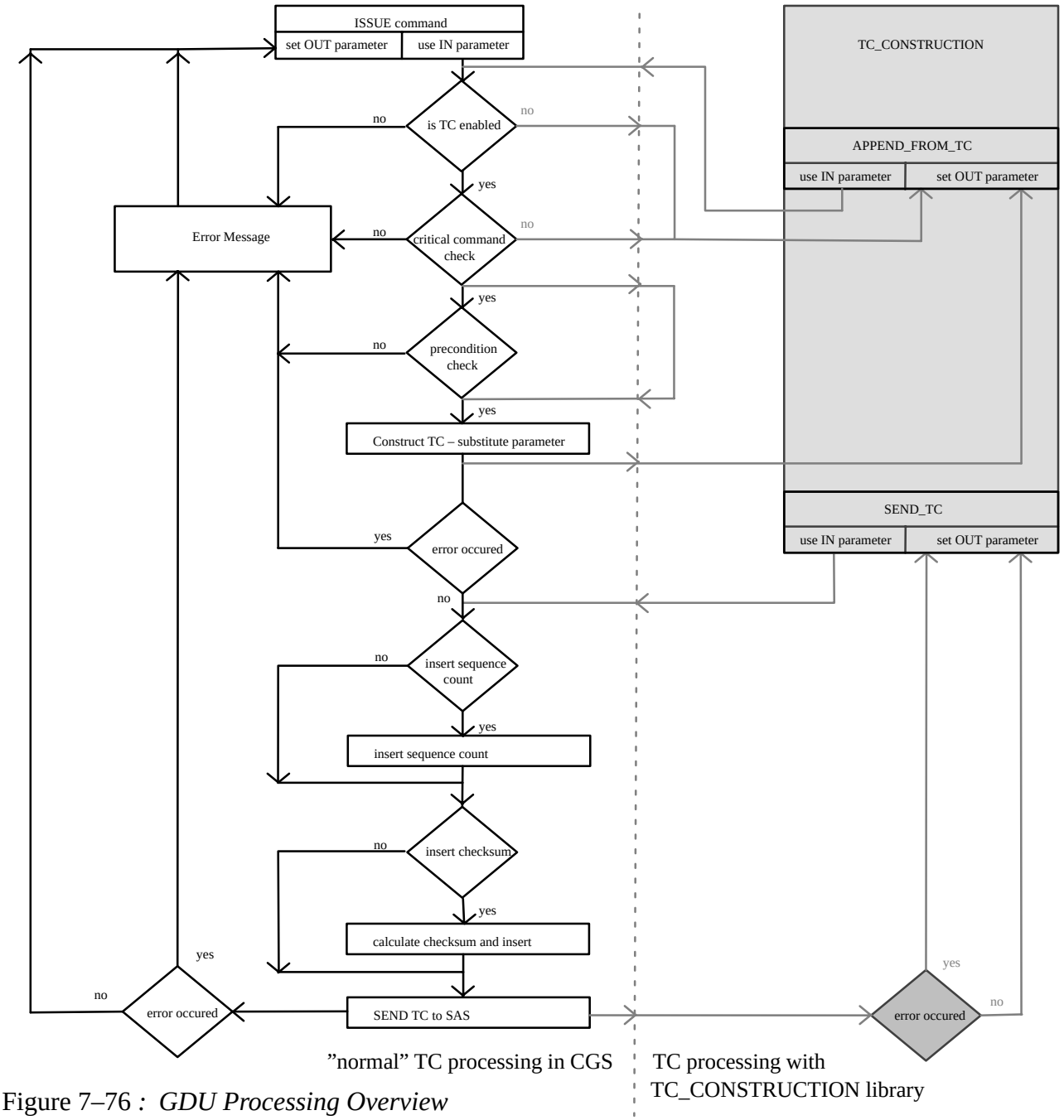


Figure 7–76 : *GDU Processing Overview*

Copyright per DIN 34

TES loads the following end item types from the MDB (LOAD_SCOE):

EGSE_ANALOG_STIMULUS
EGSE_DICRETE_STIMULUS
EGSE_BINARY_PACKET
EGSE_PREDEFINED_TC
PUS_PACKET
SWOP_COMMAND (specific for the COF project)

All these end items are translated to GDUs before sending them to the SAS specified.
Stimuli allow to send single values, while the remaining allow to send a packet of values: either in a binary buffer or as a CCSDS packet. (EGSE_PREDEFINED_TC, PUS_TC, SWOP_COMMAND).

TES generates the actual **packet** from the information given in the MDB (CCSDS Header, CCSDS Secondary Header , Predefined Data). The packet is completed by parameter given online in the UCL ISSUE command depend on the defined parameter list. The parameter values are inserted into the packet at specified places, overriding possibly the predefined values.

A GDU is **sent to the SAS** and needs to be acknowledged by the SAS. If the acknowledge is not received within a configurable timeout value, an error message is generated and the failure is indicated to the caller of the ISSUE statement.
The acknowledge code, the SAS acknowledge code (if given – see Appendix F-2 on page F–30 – ACKNOWLEDGE_COMMAND) and the SAS acknowledge time (if given) will archived togeteher with the GDUitself.

For SWOP_COMMANDs, a specific return package (response packet) may be specified. For description refer to the CGS ICD and the Ground_to_Onboard Library given in Appendix I of this document.
SWOP–COMMANDs are converted to GDUs, and response packets are handled as normal ADUs. Thus they are archived as any other ADU/GDU. The configurable parameter TES.KERNEL.SW_CMDER.ISSUE_TIMEOUT_WHEN_NO_DELAY in the *configuration.xml* allows to specify a GDU timeout for software commands in case the timeout from the UCL command is set to zero (meaning no response packet is expected).

Any sending of a GDU is logged as an event. Any GDU is archived, if archiving is enabled. The Acknowledge return status and the Acknowledge time are archived too (see Appendix F – ACKNOWLEDGE_COMMAND).

End item Grouping : GDU lists

It is possible to define GDU lists that are a list of individual GDU's. When issuing a GDU list, the test node software will issue one GDU after the other from that list.

Priority

GDUs can be issued via a UCL command with a low or a high priority. A third "emergency" priority exists but is not available to the user via the UCL command and is reserved for the GDUs to be issued as a result of a monitoring exception.

For every SAS to which GDUs are sent, three GDU queues are existing (low, high and emergency) and they are treated according to their priority.

CCSDS time

For the CCSDS time that is used in the CCSDS commands, an epoch start time can be specified in the *configuration.xml*. The three configurable parameters *TAI_EPOCH_START_YEAR*,

Copyright per DIN 34

TAI_EPOCH_START_MONTH and *TAI_EPOCH_START_DAY* in group *TES.KERNEL.GDU.TABLE* can be modified inside that file prior starting the system.

CCSDS Sequence counter

Every testnode manages the CCSDS sequence counter to be set for CCSDS telecommand, depending on the application id of the primary header. The configuration parameter in the *configuration.xml* TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID is used to indicate if the packet type field of the CCSDS primary header has to be taken into account for the application ids. By default that field is taken into account.

It is possible to define a default value for the the first sequence count value via *configuration.xml TES.KERNEL.GDU.INITIAL_SEQUENCE_COUNT* parameter.

Online Modification

The following modifications are possible online during test execution without regeneration of Test Node (SCOE) files:

- It is possible to modify the reference of the SAS to which the GDU is to be sent using a UCL command (”routing”).
- It is possible to modify the Application ID (APID) in the CCSDS header of a CCSDS Telecommand
- It is possible to modify the sequence count per Application ID (APID)
- The timeout value and the priority for a GDU can be specified online together with the ISSUE command when sending the GDU.
- The onboard time tag as well as the ground time tag can be specified online together with the ISSUE command when sending the GDU.

The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup)

7.3.3.3.1 Prevent Command Sending

It is possible to save telecommands (GDU’s) against sending. Each single telecommand will be defined in the mission database together with a flag “Inhibited for Sending”. If this flag is set to TRUE, this telecommand is not allowed to send. You can change this flag interactivly with system library calls ENABLE_ENDITEM and DISABLE_ENDITEM.

7.3.3.3.2 Critical Commands

It is possible to save telecommands (GDU’s) against unauthorized sending. Each single telecommand can be defined in the mission database together with a flag “Critcal command” and an optional password. This password is a string of up to eigth printable ASCII–characters.

If the telecommand is marked as “Critcal command”, each sending must be authorized by confirm (no password defined) or authorized by password. The user is requested in HCI to allow the sending of telecommand. Three tries are allowed for each password request. If no HCI is connected to TES, the sending of critical commands is rejected.

All unauthorized tries are logged to error message handler.

All other telecommands (without authorization flag) are handled as authorized.

Copyright per DIN 34

7.3.3.3.3 GDU Preconditions

It is possible to save telecommands (GDU's) against unnessary sending. Each single telecommand can be defined in the mission database together with preconditions. Each precondition consists of an end item reference, an operator (equal, not equal, in range, less than, less equal, grether than, grether equal) and reference values.

If the telecommand is marked as a command with precondition checking, each condition is checked before sending. If all checks are successful, then the GDU is send.

All unsuccessful tries are logged to error message handler and into TRDB.

All other telecommands (without precondition definition and with invalid precondition definition) are handled like precondition fulfilled.

7.3.3.3.4 Telecommand Verification

It is possible to defined in the MDB automatic verifications to apply after having sent GDU of the following types :

EGSE_ANALOG_STIMULUS
EGSE_DICRETE_STIMULUS
EGSE_BINARY_PACKET
PUS_PACKET
EGSE_PREDEFINED_TC
SWOP_COMMAND

This definition includes two times, the activation delay and the verification timeout as well as a set of measurements, software variables or derived values to be verified against a given value or range.

After the GDU is successfully issued, the test node will start the telecommand verification after the specified activation delay. It will then check the value of the measurements, software variables or derived values against the specified value or range. In case all checks are successful, the TC verification is then successful. In case the verification is not successful, every time the value of the measurements, software variables or derived values is modified, the checks are made. If after the specified verification timeout, the verification is still not successful, it has then failed.

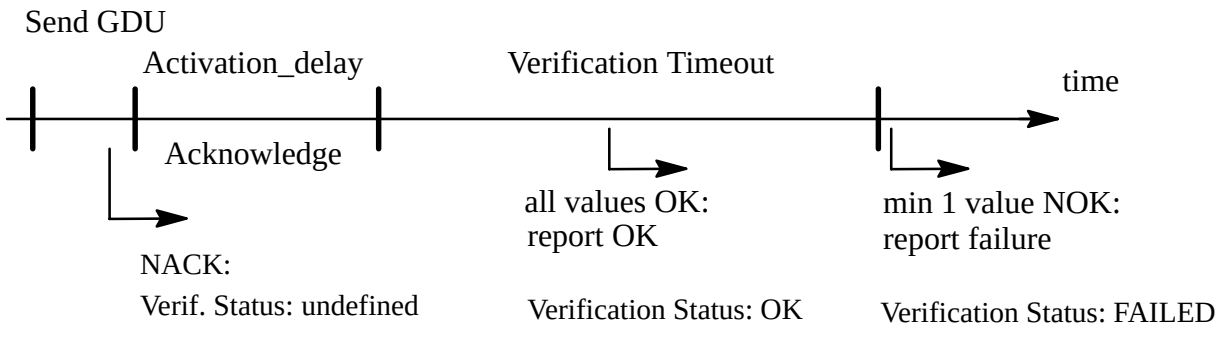


Figure 7–77 *: GDU Verification*

When issuing a GDU via GROUND_LIBRARY.ISSUE, the TC verification will be asynchronously triggered if defined, ie the UCL Ground library routine used to send the GDU will not wait until the verification is done.

Copyright per DIN 34

A UCL Ground library routine allows to check the verification status of a GDU.

A specific UCL Ground library routine ISSUE_AND_VERIFY allows to issue a GDU overwriting the specified times (activation delay and verification timeout). This operation can as well be used in a synchronous mode where it completes only at the end of the TC verification.

In *configuration.xml* the TC verification of all telecommands can be disabled by setting parameter: *TES.KERNEL.GDU.HANDLER.TC_VERIFICATION_DISABLED* to true

7.3.3.4 Archiving and Logging

Archiving
TES archives the following items into archive files:

- ADUs (Unstructured, Structured, CCSDS Packets)
- GDUs (Stimuli, Telecommands, Binary Packets)
- Changes in the SMT
- Requests for ADUs

Archiving may be enabled/disabled for the whole test node. Archiving cycles (i.e. periods after which the archive file is closed and transferred to the TRDB) may be specified when enabling the archive.
Archive files may be closed on request (CLOSE_ARCHIVE).

Archive Files can be evaluated later or during the test session using the TEV tool, generating a dump of the ADUs/GDUs in hexadecimal or ASCII format, or by recalibrating measurement values from the ADUs.

Archiving files are the basis for the replay of test sessions in the REPLAY mode.
Archive Files when closed are transferred to the TRDB and handled as part of the test session.

Command History
Additionally TES log the following items into an external command history depend on existence of this application and the setting of configuration parameter *TES.KERNEL.LOG.IN_COMMAND_HISTORY,* which default value is False:

- GDUs (Stimuli, Telecommands, Binary Packets)
- Response packets for SWOP commands

Event Logging
Any user related action in TES is logged as an event in the Event Log of the TRDB. User Messages are sent in addition to the HCIs connected to the TES, and thus are logged in the log files of the message handler.

Event Logs can be evaluated by the TEV tool after or during the test session to generate complete or selected event lists. Message Logfiles can be viewed by the message handler during or after the ongoing test.

Engineering Value Logging
TES will **log** each new engineering value if defined so for this end item and if the value is different from the previous one. Enabling/disabling of engineering value logging (EVL) is possible via UCL commands, either on individual end items or on virtual trees / monitoring lists. EVL can be used later during evaluation to generate datasets and then data listings, graphical curves or Excel spreadsheets.

7.3.3.5 UCL Execution

UCL Language

Copyright per DIN 34

UCL is defined as a set of Test object or Test system oriented commands together with a set of control commands allowing specification of automatic procedures with conditional execution. (UCL is used for both Onboard and Ground purposes).

The UCL Language

- Provides type conversions
- Supports engineering units
- Supports CGS pathname concept
- Conditional statements
  – IF, CASE, LOOP, WHILE, etc
- Libraries
  – user libraries and system libraries

UCL allows for definition of Automated Procedures (APs) that may be activated by an interactive command and can call other APs. Each AP is compiled offline during test preparation by the UCL Compiler. This generates an intermediate language format called I–Code. This is interpreted by the UCL Interpreter called by the Test Execution Software.

UCL allows to define User Libraries implementing additional procedures that can be called from any AP or HLCL command window. User Libraries are loaded to a test node. and executed as I–Code.

UCL system libraries are specification of services/procedures implemented in TES. System library specification must be imported into APs or User Libraries to allow them to be used.

Different UCL System Libraries provides a broad range of function for following topics (for details see appendix I on page I–1):

- **Data Processing** control
  – Start/Stop acquisition of end items
  – define conditions to control processing
  – get status on conditions
  – get acquisition status
- **Monitoring** control
  – change online limits
  – enable/disable monitoring
  – get monitor status
- **Time** management
  – get local time and simulated mission time (SMT)
  – SMT management
  – wait_until, delay
- **Automated Procedure** control
  – start another automated procedure (asynchronous, synchronous)
  – suspend/resume automated procedure
  – get status of automated procedure
  – exchange messages between automated procedures
- **SAS** control
  – start / stop a SAS

Copyright per DIN 34

– change SAS current mode (init, reset, etc.)
– get status of SAS
– exchange messages with SAS (synchronous, asynchronous)

- **Stimulus(GDU)** generation
  – send stimulus/TC
  – send a list of stimuli/TC
  – enable/disable certain stimuli/TC
- **Log** & **Event** handling
  – generate log events
  – generate user events
  – enable/disable engineering value logging
- **Archive** control
  – enable / disable archiving
  – define archiving cycle
  – close the archive
- **Synoptic** display control
  – show a dedicated synoptic on a dedicated screen
  – remove a synoptic from the screen
- **User input & output**
  – write a message to the user
  – read a message from the user
- Control of **Conditions**
  – define/delete conditions online
  – enable/disable conditions
  – interrogate condition status
- General **conversion** procedures and functions

Refer to Appendix I for a specification of the Ground Library.

The writing to **text files** and reading from text files is implemented via a specific user library (**FILE_IO_LIB**). Refer to Appendix M for further description.

CGS System Libraries are:

```
* CPL_Library       cpl_library_.ucl       Library to be imported in any
                                           CPL script.
                                    Body ID: any

* FWDU_Library      fwdu_library_.ucl      Library imported in any FWDU
                                           command table.
                                    Body ID: any

* Ground_Common      ground_common_.ucl      System interface to CGS/VI-
COS.
```

Copyright per DIN 34

General definitions (types/ constants)
Body ID: any

* Ground_Library ground_library_.ucl System interface to CGS/VI-COS.
Body ID: 2

* Ground_To_OB_Lib ground_to_ob_lib_.ucl Ground interface to send com-mands
to the onboard system.
Body ID: 4

* Ground_Values ground_values_.ucl Ground access to measure-ments,
calibration and decalibration
Body ID: 8

* Grd_Conversion grd_conversion_.ucl General Conversion Functions for
Ground Checkout
Body ID: 11

* Math_Lib math_lib_.ucl Mathematical library based on type REAL.
Body ID: 6

* Math_Lib_Long math_lib_long_.ucl Mathematical library based on type LONG_REAL.
Body ID: 7

* Onboard_Library onboard_library_.ucl System interface to the Co-lumbus
onboard system.
Body ID: 3

* Raw_Data_Library raw_data_library_.ucl Defines procedures to re-trieve
data from raw TM packets (e.g. from
Packet_Library) via UCL
Body_Id:12

Copyright per DIN 34

```
* Packet_Library     packet_library_.ucl     Defines procedures to main-
tain
                                            TM packetbuffers and Trigger
APs
                                        Body_Id:13

*  TC_Construction      tc_construction_.ucl      Defines  procedures  to
construct
                                          a Telecommand via UCL
                                       Body Id: 9*

* ATV_Library        atv_library_.ucl        Defines Procedures to issue
nested
                                              TC's, to get/set and sequence
count
                                       Body_Id:14
```

AP Execution

Up to 40 APs may be executed in parallel on each test node. Each AP is executed within one slot of the AP Executor. By default only 20 APs can be executed in parallel. The number can be decreased down to 1 / increased up to 40 by modifying the configuration parameter *TES.KERNEL.UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER* in the *configuration.xml* prior starting the system. It must be noted that increasing the number of APs that can be executed in parallel leads to an increase of memory required by the test node software as for every slot, memory must be allocated for loading and executing the AP.

By default, 5 slots are reserved for so called emergency APS. i.e. APs that are started by the monitoring/ condition function. In case a emergency AP is running, all other non–emergency APs are suspended. This number of slots reserved for monitoring exception can be modified in the *configuration.xml*, parameter *TES.KERNEL.UCLI_CONTROLLER.MAXIMUM_NUMBER_OF_EMERGENCY_APS* prior starting the system.

All APs on one testnode can be aborted by single HLCL command ABORT_ALL_APS or one single AP by ABORT_AP command.

APs can be suspended and resumed for execution. The actual status of the AP can be monitored in the AP Status window.

If defined so in the *configuration.xml*, TES keeps the actual statement of an executing AP in a specific HK value, which can be monitored in the AP status window. Furthermore, the executed I–Code can be logged in a specific file, as defined in the *configuration.xml*. This allows minimum AP debugging support during AP development. For debugging purposes use the UCL debugger.

A CGS User involved in Testing operations will be able to issue **interactive commands**, from the Workstation, to distributed test software via appropriate User interaction methods such as windows, menus, dialog boxes etc. These commands encompass UCL statements and other Workstation or Test specific commands which together form the High Level Command Language (HLCL).

Copyright per DIN 34

The same AP can be started several times in parallel, except for the AP's started as result of a monitoring exception or as result of a condition, that will only be started if not already running.

Priority

AP's can be started via a UCL command with a low or a high priority. A third "emergency" priority exists but is not available to the user via the UCL command and is reserved for the APs to be issued as a result of a monitoring exception.

The scheduling of the AP's on one test node is an optimised preemptive scheduling model. All AP's of the same priority level are executed in parallel. Whenever an AP of a given priority level is running, AP's of lower priority levels are suspended. They are automatically resumed when no AP of a higher priority level is running any more.

Online Modification

UCL APs and user libraries may be reloaded from the MDB after re–compilation without re–initialising the test node. This is done using a Ground library routine. Thus modification of APs during online testing is supported. The modifications are valid until the test node is reinitialised with forced reloading (see Test Setup).

7.3.3.6 Communication with SAS

Test Nodes may communicate with a set of SAS, running on the same node or on remote nodes.
SAS may receive GDUs or may deliver ADUs. SAS also may write to software variables or read raw or engineering values from the testnode.

SAS can be controlled interactively or from APs/HLCL Sequences, using the command implemented in the UCL Ground Library.

The following limitations exist:

- Up to 20 SAS can be active for the same test node.

Different configuration parameters in the *configuration.xml* can be used to tailor the test node to the system needs. Timeout values can be specified for the different commands sent from the test node to the SASes. Those configuration parameters are :

TES.KERNEL.SAS.ENABLE_ACQUISITION_TIMEOUT,
TES.KERNEL.SAS.DISABLE_ACQUISITION_TIMEOUT,
TES.LKERNEL.SAS.OAD_APPLICATION_TIMEOUT,
TES.KERNEL.SAS.INIT_APPLICATION_TIMEOUT,
TES.KERNEL.SAS.START_APPLICATION_TIMEOUT,
TES.KERNEL.SAS.RESET_APPLICATION_TIMEOUT,
TES.KERNEL.SAS.GET_APPLICATION_STATUS_TIMEOUT,
TES.KERNEL.SAS.WRITE_MESSAGE_TO_APPLICATION_TIMEOUT,
TES.KERNEL.SAS.DOWNLOAD_FILE_TO_APPLICATION_TIMEOUT,
TES.API.CONTROLLER.TIMEOUT_FOR_READ,
TES.API.CONTROLLER.TIMEOUT_FOR_SEND,
TES.API.CONTROLLER.TIMEOUT_FOR_RESPONSE,
TES.API.CONTROLLER.DEFAULT_RETRIES and
TES.API.CONTROLLER.TIMEOUT_PERIOD_FOR_READ_CMD.

7.3.3.7 Replaying Data

In Replay mode, the data is replayed from the archive files produced during a test and specified during the replay setup:

Copyright per DIN 34

- ADUs (Unstructured, Structured, CCSDS Packets, PUS_Packets)
- GDUs (Stimuli, Telecommands, PUS_Packets, Binary Packets)
- Changes in the SMT
- Requests for ADUs
- Suspension for ADUs

The data from the event log and engineering value log are not replayed. Neither the AP's and sequences that have been started during the test are replayed.

When an ADU request is detected that was a start acquisition command, all measurements from that ADU will be processed from that ADU.

When an ADU request is detected that was a stop acquisition command, the acquisition of all measurements from that ADU will be stopped.

When an ADU is replayed from an archive file, the measurements it contains that are acquired will be processed, as during the normal mode (see Monitoring and Data Processing). The measurements can be acquired by replaying a start acquisition command or using the UCL commands as during the normal mode. Note that as it can occur that the start acquisition commands are not replayed (e.g. due to a restriction of the time frame to replay), a configuration parameter in the *configuration.xml* allows to automatically enable all measurements in the test nodes to avoid having to enable everything via the UCL command. That configuration parameter is called *TES.KERNEL.DATA_PROCESSOR.ENABLE_ALL_ADUS_IN_REPLAY* and must then be set to true prior starting the system.

When a GDU is replayed from an archive file, a message will be issued to the user. In case this would lead to a too large number of messages and this is not wanted, the generation of the messages can be prevented by setting the configuration parameter *TES.KERNEL.REPLAYER.GDU_MESSAGES* to false in the *configuration.xml* prior starting the system.

When a change of the SMT (start or stop SMT command) is replayed from an archive file on the MTP, the SMT will be accordingly set during the replay session.

The test node is managing the replay time according to the replay time frame and the speed factor specified during the setup phase. As the SMT is a system wide time, in case the speed factor is different from 1, in opposition to the internally managed replay time, the SMT value will "run" at a speed factor of 1. The MTP will then cyclically "re–synchronise" the SMT to the replayed value in case of deviation. This synchronisation will be performed by default every 5 seconds. This value can be modified in the configuration parameter TES.KERNEL.REPLAYER.SMT_UPDATE_PERIOD_IN_REPLAY of the *configuration.xml* prior starting the system.

During replay mode, all other functionality in the test node are available as during normal mode, i.e. AP's can be executed, monitoring can be done, etc.

A few differences are in the fact that some UCL library routines will be ignored, as for example the starting or stopping of SMT, the generation of GDU, the enabling or disabling of archiving.

7.3.3.8 Simulating Data

In simulation mode, the test node has the same functionality as in normal mode except that the ADU's are not acquired from a SAS and the GDU's are not sent to a SAS.

The sending of a GDU will result in the generation of a message.

Copyright per DIN 34

In simulation mode, the ADU's are simulated. ADU that will be required to be simulated must have in the MDB a simulated ADU description. UCL commands allow in simulation mode to set specific raw value (end item values) of a structured ADU or to set specific data in an unstructured ADU or in a CCSDS packet.

Copyright per DIN 34

7.3.3.9 Internode Communication

On test nodes, data may be acquired from other nodes by simply referencing end items loaded to those nodes. TES manages to route the data to the actual node.

The following is supported:

- Starting of APs on remote nodes
- Reading of end item values (measurements, software variables or derived values) from remote nodes
- Writing Software variable from remote nodes
- Sending GDU (Stimuli) or GDU lists via remote nodes
- reading the TC verification status from a remote GDU

Starting an AP remotely is performed by specifying on which node the AP has to be started while the remaining remote operations are transparent (any test node knows which test node is maintaining a given measurement, software variable, derived value or GDU).

The following limitations exists:

- Monitoring attributes cannot be interrogated nor set on remote nodes
- Acquisition, Monitoring, Logging and Archiving can be enabled only on local nodes
- APs can communicate only to SAS having been started and assigned to the local node.

In order to be able to have internode communication, each testnode must maintain references of the data maintained by remote nodes (references to measurements, software variables, derived values, GDUs and GDU lists). The test nodes synchronise themselves automatically during the test setup. The references are held in a table in each test node. The length of that table has been defined to be able to have up to 20000 external references by default. This value can be adapted to special needs in case a system is maintaining more data by modifying the parameter *TES.KERNEL.DISTRIBUTION_TABLE.MAX_NB_SID* in the *configuration.xml* prior starting the system.

Copyright per DIN 34

7.3.3.10 The configuration of TES

TES behaviour can be controlled by various configuration parameters defined in the file *configuration.xml*.

The file has many parameters for the group TES. In the normal case there is no need for changing the parameters, because appropriate default values has been defined, except for 2 parameters (TES.KERNEL.SW_CMDER.EXEC_FLAP_SWOP_PATHNAME and TES.KERNEL.SW_CMDER.EXECWAIT_FLAP_SWOP_PATHNAME) that are specific for the COF project. However an 'advanced' user may want to change some of the parameters to adapt the TES behaviour to his needs. Many of the parameters has only interest for the engineers and the developers of the system (marked with 'E'). Some of them, marked with 'U', the advanced user should know about. In the following the most important of them are described, also those marked with 'E'.

The remaining parameters, not described here must not be modified.

The TES part of CGS configuration (configuration.xml) file and its location are described in Appendix K.

Copyright per DIN 34

8 TEST EVALUATION

8.1 General

CGS allows for the evaluation of data stored in the Test Result Database (TRDB) during a running test session or after having executed a test session with the Unit Under Test (UUT) or with simulated parts of it.

The data archived often needs further evaluation in an off–line session, especially to verify whether data generated by the UUT is in the required margins or to analyse non–nominal situations during a test.

In the beginning of a TEV session the user must define the data area in the Test Result Database (TRDB) to be evaluated. This means providing the name of test session(s) for which data shall be evaluate and eventually a time frame.

During the evaluation work the user frequently produces output files, held by default under the user's account. These output files are the so–called evaluation result files and the definition files.
The user may also specify an Evaluation Session, under which he can store the evaluation result files in the TRDB. This is necessary for final archiving or exporting the result files.

Finally, the user may convert Data Set and Data Listing into files in comma separated value format (csv) , in order to proceed the evaluation with Excel. (see Chapter 8.5.1 File Handling).

The following basic steps describe the general proceeding to be performed in every TEV session:

- Select the test session(s) to evaluate
- Select a Configuration Unit of the Mission Database (CCU) containing the test data (optional : by default, the one of the selected test session)
- Select or create a test evaluation session (optional)
- Select the appropriate TEV tool for data evaluation
- Generate a Data Set, necessary for generating :
  - Data Listing
  - Statistics (in CGS 6.3.2 replaced by Data Viewer)
  - Graphs (in CGS 6.3.2 replaced by Data Viewer)
- Store the evaluation results (optional) (the evaluation results are by default stored in the tev working directory)
- Convert Data Set and Data Listing in csv format, in order to proceed to an evaluation through the Excel Tool.

The On–line Help provided by TEV is based on the help features provided by the OpenWindows environment.

From the Help key on the keyboard, the user can request information about any TEV window. When this key is pressed, a Help "push pin window" is displayed containing textual information about the window item pointed by the cursor.

TEV can also be started in batch mode.

8.2 Changes in CGS 6.3.2

In CGS 6.3.2 the original Statistics Generation tool and Graphics tool have been replaced by the new Data Viewer tool.. By default, these tools don't appear any more in the **Tools** menu of the TEV Main window and

Copyright per DIN 34

the **Result Type** menu of the File Manager window. However, TEV provides a backward compatibility mode allowing to access and use these tools in the same way as in prior CGS versions. This is achieved by setting the evironment variable **TEV_PRE_CGS_632** to **TRUE**.

```
% setenv TEV_PRE_CGS_632 TRUE
```

The new Data Viewer tool is invoked from the Data Listing tool.. It is opened on data listings generated in CSV format. The CSV format has now become a TEV internal format, so data listing result files are stored in the subdirectory ***RESULTS/DATA_LIST*** under the tev_working_directory ***user_home/.cgs/tev*** instead of in the subdirectory ***EXTERNAL/DATA_LIST***.

TEV in batch mode is currently not affected by these changes, i.e. all tools are still available as before without explicitely activating backward compatibility mode.

8.3 Changes in CGS 7.1.0

The button labeled ”Convert...” that allowed to save a data set in CSV format has been removed from the File Manager Tool. Use the Data Listings Tool to generate a data listing in CSV format from a data set instead.

The Data Set Generation Tool allows to select the parameters to be included in the data set from the list of all parameters that had been logged in the EVL during the currently selected execution sessions.
It is now possible to configure which time tags (i.e. from archive, ADU or CCSDS packet) should be used in the data set for local time and mission time.

In the Data Listing Tool, the Save... operation of the Result menu has been renamed to Save Current.... An additional operation Exec & Save... allowing to generate a data listing and save it in one step has been added to the Result menu.

8.4 Changes in CGS 7.3.0

TEV in interactive mode now stores all definitions in ASCII format.

The Event Logging Tool has been adapted to the new unified CGS event/message format. However, definitions in pre–CGS 7.3.0 ASCII format are still supported.

Event listings can now be generated in CSV format. TEV opens the CGS Message Handler window to view such results.

The restrictions on the number of parameters in data sets and data listings (50 resp. 10 parameters at maximum) have been removed.

The restrictions on the number of selectable ADUs and GDUs in the Raw Data Dump Tool (20 ADUs + 20 GDUs at maximum) have been removed.

The restrictions on the length of definition and result names (20 characters at maximum) have been removed. However, the length of an absolute pathname of a definition or result file is restricted to 256 characters at maximum.

The Final Archive Tool has been removed, use the Export/Import Tool instead.

The Export/Import Tool does no longer support exporting partial execution sessions, export complete execution sessions instead.

8.5 Changes in CGS 7.3.1

The print operations on definitions and results have been removed, so the *Definition* menu’s *Print...* and *Print Current* operations as well as the *Result* menus’s *Print File...* and *Print Current* operations are no longer available.

Copyright per DIN 34

8.6 Using TEV in Batch Mode

TEV provides a batch utility through which the user may execute the logging event tool, the raw data dump tool, the data listings tool, the data set generation tool and the graph in one shot tool (this last tool is only available in batch mode). The TEV execution conditions are defined in the input parameters of the batch command. Each batch process started executes one batch command.

The user enters the batch command from the command line. This invokes the start script which recognises that batch TEV is to be started. The input parameters are passed to the start procedure.

It is only possible to run one instance of TEV at a time on a workstation. Multiple instances of interactive TEV and batch TEV may be run on different stations.

8.6.1 Inputs

Four inputs are required to invoke batch TEV :

- a tool identifier to specify the tool to be executed.
- a definition file containing the definition to be executed (ASCII text, optional for EVENTS LOGGING TOOL and RAW DATA DUMP TOOL). The syntaxes to be used for definitions are described in remplates files provided in the directory ***$CGS_HOME/etc/data/tev***. See Appendice K for the description of the templates files.
- a session file containing the execution sessions names with which TEV is to be initialized (and the CCU information if different from the one of the session(s))
- the name of the result file where the result is to be written.

One additional input may be provided for the generation of Data Listings :

- the name of the format : CSV resp. EXCEL, TEXT resp. ASCII, ADT or LACIS

One additional input may be provided for the generation of Event Logs :

- the name of the format : CSV resp. EXCEL, TEXT resp. ASCII or ADT

One additional input may be provided for the generation of Data Sets :

- the name of the format : CSV resp. EXCEL or ADT

This option is not necessary if the definition already included a format option. This option –fo allows the user to use several times the same definition, generating several results in different format. By default Data Listings are generated in ASCII and Data Sets in ADT (Internal TEV binary format).

The inputs may be supplied in any order but the following syntax should be used :

**start_tev –to <tool_name> **
**–se <session_file_name> **
–de <definition_file_name> \ (optional for RAW_DATA and LOG_EVENT)
**–re <result_file_name> **
– fo <format>

Tool Name :

The tool name specifies the TEV tool which is to be executed and must be one of the following :

Copyright per DIN 34

- **LOG_EVENT** for executing the log event tool
- **RAW_DATA** for executing the raw data dump tool
- **DATA_SET** for executing the data set tool

  LISTING for executing the listing tool
- **ONE_SHOT_GRAPH** for executing the one shot graph tool (only avalaible in batch mode)

8.6.2 Session File

The session file contains the sessions with which TEV is to be initialised and the CCU information. It is an ascii file and must exist in $VICOS_TEV_WD/SESSIONS. It must be of the following format:

SESSION EXEC_SESSION_01
SESSION EXEC_SESSION_02
SESSION EXEC_SESSION_n
SESSION DEFAULT_TEST_SESSION

Optional :

CCU_CONFIG_NAME Valid name of an existing CCU in MPS.
CCU_VERSION Valid version number of CCU in MPS.
CCU_PATHNAME Valid pathname of an existing CCU in MPS.

The following old format is also supported and can be used together with the new one :

SESSION: EXEC_SESSION_01
SESSION: EXEC_SESSION_02
SESSION: EXEC_SESSION_n
SESSION: DEFAULT_TEST_SESSION
CCU_CONFIG_NAME: Valid name of an existing CCU in MPS.
CCU_VERSION: Valid version number of CCU in MPS.
CCU_PATHNAME: Valid pathname of an existing CCU in MPS.

This file may contain a maximum of 10 execution session names plus the *default_test_session*. The *default_test_session* may appear at any position in the list of sessions.

The sessions with which TEV is initialised will be allocated to the batch user for the duration of the job.

The mission name, element configuration name and system tree version number are obtained by TEV from the session(s).

If any or all of the ccu configuration name, the ccu version or the ccu pathname are missing or wrong then these are obtained by TEV from the session(s). If the sessions(s) have different CCUs, TEV takes the one from the most recent session.

8.6.3 Definition File

The definition file contains the definition to be executed :

- create the definition file with TEV Interactive, using the menu ***"Definition:Save"*** in the appropriated tool. If the definition is saved while ***"Select Initial Time Frame"*** is selected,

Copyright per DIN 34

the definition will consider all data of the session(s). Otherwise the ***"Selected Time Frame"*** is part of the definition.

- create the definition file with a text editor : the definition file must follow a specific syntax described in specific syntax files. (see Appendice K). The ASCII definition file must exist in the appropriated definition directory ie.
  – $VICOS_TEV_WD/DEFINITIONS/EVENT_LIST or
  – $VICOS_TEV_WD/DEFINITIONS/RAW_DATA_DUMP or
  – $VICOS_TEV_WD/DEFINITIONS/DATA_SET or
  – $VICOS_TEV_WD/DEFINITIONS/DATA_LIST or
  – $VICOS_TEV_WD/DEFINITIONS/ONE_SHOT_GRAPHS

8.6.4 Result File

Once the definition has been executed the result is written in the result file which is created in the appropriated directory :
– $VICOS_TEV_WD/RESULTS/EVENT_LIST (TEXT, CSV, or ADT (binary) format) or
– $VICOS_TEV_WD/RESULTS/RAW_DATA_DUMP (TEXT format) or
– $VICOS_TEV_WD/RESULTS/DATA_SET (ADT (binary) format) or
– $VICOS_TEV_WD/RESULTS/DATA_LIST (TEXT, CSV, ADT (binary) or LACIS format) or

– $VICOS_TEV_WD/RESULTS/ONE_SHOT_GRAPHS (PostScript format)

ASCII and ADT formatted result files may be also displayed using the appropriated interactive TEV tools. ADT formatted result files can also be read using the tev_api library.

Note: In batch TEV, confirmation to overwrite an existing result file is not possible.
Existing result files will be overwritten by batch TEV.

8.6.5 Error Handling

All errors will be logged to the CGSI and also written to stdout. Successful completion will be written to stdout. Errors will be logged if input parameters are missing.

To log messages and errors in a file the user should redirect the output using standard unix commands.

Copyright per DIN 34

8.7 Test Evaluation Preparation

The *Test Evaluation Software (TEV)* is invoked by the main CGS Task Selector as described in chapter NO TAG.

Precondition for executing the TEV software is that all basic CGS processes have been started.

Test sessions can be defined by the TSCV software. Recorded test sessions can be retrieved and evaluated by the TEV software.

Test Evaluation Software starts by displaying the main window that contains the main menu buttons. These buttons activate the TEV operations for the SESSIONS, UTILITIES and the EVALUATION TOOLS.

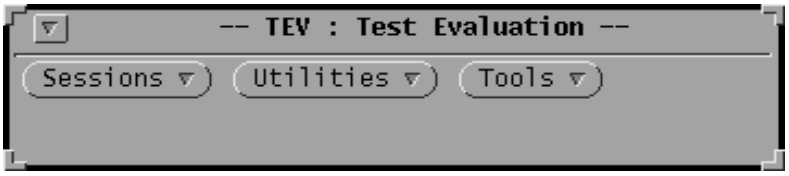


Figure 8–1 : *TEV Main Window*

The **Sessions** menu button allows the user to :

- select an Evaluation session to store TEV result files in TRDB or retrieve TEV result files from TRDB,
- to delete a given Evaluation session,
- to select Execution sessions and the CCU,
- to delete a given Execution session,
- to export and import sessions to/from the Final Archive Medium

The **Utilities** button allows the user to :

- start the file manager, for handling of evaluation result files and external files in the working directory.
- start the data sets merger utility to merge two data sets.
- start the data set/events listing merger

The **Tools** button allows the user to start the Test Evaluation tools.

The message **Start TEV : OK** is the status message reported to the user in the main window after TEV has been successfully initialisated. This information line has two parts separated by a colon(:): the operation executed and its returned status. The status may be Ok or an error message. For each tool main window in TEV, context driven messages will appear in the footer.

- *When starting TEV, some errors may arise, like a DBS connection problem, missing variables in the environment.... For more information about the TEV error report service refer to Appendix D.*
- *Note that the main window buttons (and some other buttons) work in two different ways :*

Copyright per DIN 34

Selecting Explicitely a Function from a Pop–Up Menu

- Move the mouse pointer on the button.
- Press and hold the right mouse button. The pop–up menu is displayed.
- Move the mouse pointer up and down the pop–up menu still holding the mouse button until you find the desired function. The function becomes highlighted.
- Release the mouse button. The desired function is activated (e.g. a new window is opened).

Activating the Default Function from a Pop–Up Menu

- Move the mouse pointer on the button.
- Click on it with the left mouse button. The default function in the pop–up menu is activated.

8.7.1 Evaluation Session Definition

An Evaluation Session has to be specified if the user wants to store the results of their data evaluation activities (i.e. the Evaluation Result files) in the TRDB.

The Test Evaluation can be performed without specifying an Evaluation Session. By default the result files are stored in a so-called 'Working directory' in the user's own UNIX directory structure : **user_home/*.cgs/tev***. The user has no possibility to set–up a mode, which would implicitely store all the result files in the current selected evaluation session. The user shall explicitely use the **Store** button from the **File Manager** Utilities.

Figure 8–2 : *The TEV **Sessions** Menu*

To access Evaluation Session operations, the user must select **Evaluation...** item in the *sessions* menu (Figure 8–2)

Copyright per DIN 34

The **TEV : Evaluation Session** window (Figure 8–3) appears:

Figure 8–3 : *Initialize Evaluation Session Window*

List Existing Evaluation Sessions

- Select a time frame to restrict the list of evaluation sessions.
- Type a name substring (containing wildcards (*)) as filter for the session names to be listed.
- Select the order in which the sessions shall be displayed (chronological or alphabetical).
- Click on the **List** button, to update the list of sessions according to the selection criteria.

By default the time frame and name substring will select all entries. This means that when evaluation sessions exist, there are by default all listed. The order by default is chronological. Note that the selection criteria are combined by a logical AND.

The user can reuse and update an existing Evaluation Session :

Copyright per DIN 34

Select an Evaluation Session

- Select a time frame to specify the evaluation sessions to be listed.
- Type a name substring (containing wildcards (*)) for selection of sessions.
- Select the order in which the sessions shall be displayed (chronological or alphabetical).
- Click on the **List** button.
- Move the mouse into the Evaluation Sessions list.
- Scroll the list by using the scrollbar if necessary.
- Click on the desired name.
- Click on the **Accept** button.

When a session name is selected in the list, any previous selection will be automatically deselected because the list accepts zero or one selection.
When a session name is selected in the list, its name is copied to the *Session Name* field and its associated information is also shown. None of these fields are editable.

- *Note that once the user has selected a session in the list, the left–hand side button has for label ”Accept”.*

The user may see the sizes of an evaluation session for each evaluation data types : for this select the session in the list and click on the **”Session Size”** button.

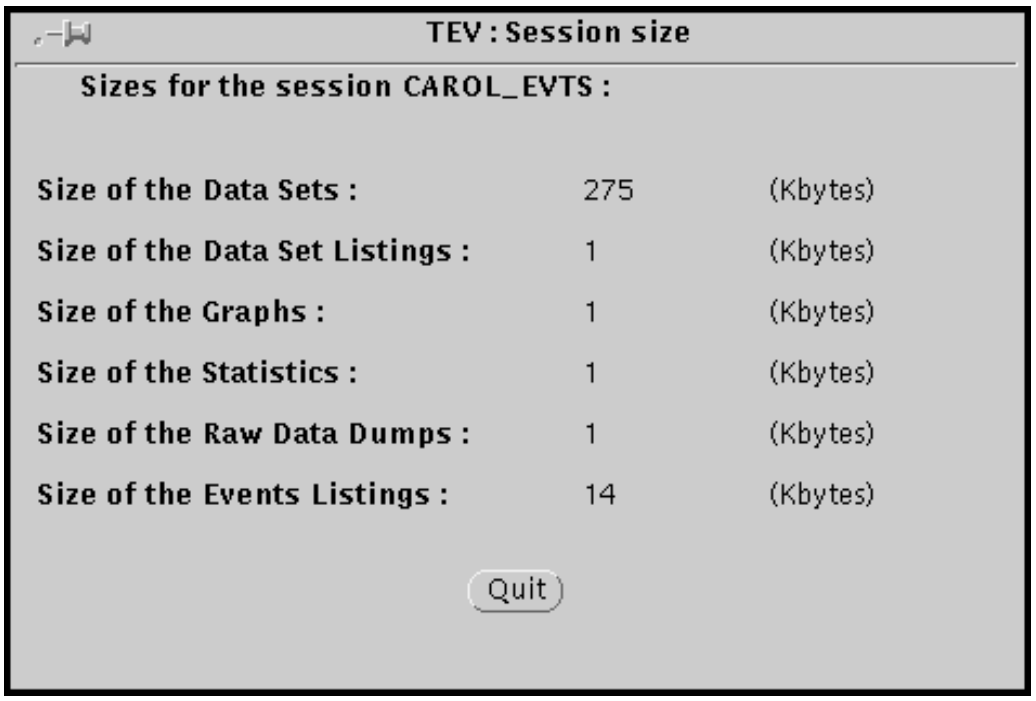

TEV : Session size
Sizes for the session CAROL_EVTS :
Size of the Data Sets : 275 (Kbytes)
Size of the Data Set Listings : 1 (Kbytes)
Size of the Graphs : 1 (Kbytes)
Size of the Statistics : 1 (Kbytes)
Size of the Raw Data Dumps : 1 (Kbytes)
Size of the Events Listings : 14 (Kbytes)
Quit

Figure 8–4 : *Evaluation Session Size Window*

For each TEV evaluation data types (Data Sets, Data Sets Listings, Graphs and Statistics, Raw Data Dumps and Events Listings) DBS may have created a directory structure. The size ”1” indicates the presence of an empty directory.

Copyright per DIN 34

Knowing the sizes of a session may be important before exporting or archiving on the MO device.

The user can create a new evaluation session. To be in creation mode, no session shall be selected into the list. In this case, the left–hand side button has for label ”Create”.

Create an Evaluation Session

- Deselect any previously selected session in the list.
- Type the name of a session (up to 20 characters) in the **Session Name:** text field.
- Type the purpose of the session (max. 80 characters) in the **Purpose:** text field.
- Press the **Create** button.

The new session appears in the list of evaluation sessions. All the information fields others than name and purpose are set–up by TEV, and not editable.

When the user either creates or accepts a session this session is allocated to him. This prevents the session from being deleted, archived or exported by another user. The allocation is cleared when the user disconnects ie quits TEV or creates or accepts another session. Other users may allocate the same session. A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not **exclusively** allocated to him.

If there is no error when accepting the selection, the Evaluation Session window is iconised. An error may be caused by an invalid session name, a missing purpose, a problem accessing the Test Result Database...

Copyright per DIN 34

8.7.2 Execution Session Initialization

The user must select one or a set of execution sessions from the TRDB, in order to generate Raw Data Dumps or Data Sets. It is not necessary to select sessions to load some already existing Raw Data Dumps, evaluate existing Data Sets or perform some merge. Nevertheless, it is at least necessary to select a CCU in order to access MDB items (for example to generate a Definition to be used in Batch Mode). A CCU is automatically selected when one session or more are selected.

Note: Execution session are described in chapter 7.2 as test sessions.

The user can also select explicitely a CCU and/or store a default CCU, which will be automatically selected as default.

The selected execution session(s) as well as the selected time frame identify the set of data which can be dumped and for which Datasets can be generated.

The test execution sessions may be selected according to time frame, name, state, location and mode selection criteria.

The Execution Sessions Initialization operations are :

- the selection of up to ten execution sessions from which TRDB data are extracted
- the selection of a CCU

Note that the **Select Execution Sessions** *window (Figure 8–5) may be displayed at any time, changes will only be accepted if no tool is running.*

To access Execution Sessions operations, the user must select the ’Execution...’ item in the *Sessions* menu (Figure 8–2) .

The **TEV : Select Execution Sessions** window (Figure 8–5) appears :

Copyright per DIN 34

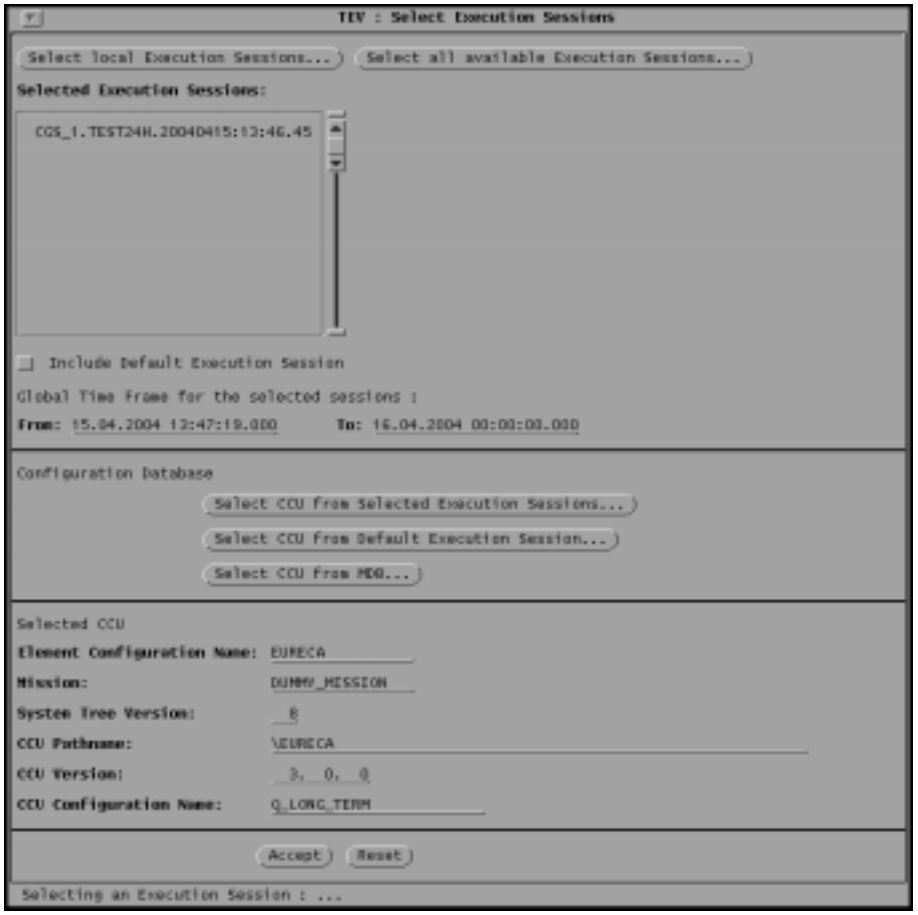


Figure 8–5 : *Select Execution Sessions*

The **TEV : Select Execution Sessions** window is split into 3 areas :

- the first area concerns the selection of execution sessions
- the second area concerns the explicit selection of a CCU (optional)
- the third area displays the information about the currently selected CCU and is read only

8.7.2.1 Select execution sessions :

General overview of the procedure to select executions sessions :

Copyright per DIN 34

Select Execution Sessions

- Press the **Select local Execution Sessions...** button to access the list of sessions from the local TRDB or
- Press the **Select all available Execution Sessions**... button to access the list of sessions from the local TRDB and the list of sessions from the TRDB, which is located on the Central Sessions Log host. (This one is described as entry in the system topology table).

*The window **TEV: Select Test Sessions** pops up.*

Use the different filters to restrict the list of sessions

*For example select **Session : State Open** to list only the opened sessions. Select **Session Location : FA Only** to list the sessions, which data are Final Archived.*

Click on the **List Sessions** button to validate the selection criteria and update the list of execution sessions available for selection.

- Select the desired Executions Sessions from the list.
- Press the **Accept** button. The selected sessions are now displayed in the **TEV : Select Execution Sessions** window.

*In the area **Seleted CCU** appears the CCU which is selected. A warning message will be displayed if the sessions selected were created with different CCUs. In this case the CCU of the newest Execution Session will be selected and the evaluation is limited to those parameters known from the selected CCU. The newest Execution Session is the session with the newest begin date.*

Click on the **Include Default Execution Session** check box in the window **TEV: Select Execution Sessions**, if the data from the default test session has to be included.

The Default Execution session contains the test logs of those test execution sessions which were started without defining an execution session by name. The default session may be selected together with other sessions in the list or alone.

The window *Select Test Sessions* functionalities :

A first area presents a set of selection criteria to reduce the list of execution sessions to be presented for selection. The selection criteria are combined with a logical AND. A click on the button **List Sessions** validates the selection criteria and displays the list of execution sessions accordingly.

The second area is the list of execution sessions presented for selection.
The execution sessions list contains the session name, the name of the user which has created the session, the session status, a status indicating wether the data are Final Archived and the session Time Frame.

Shift the small knob icons to the right or to the left to make appear accordingly the different columns of the list.

Copyright per DIN 34

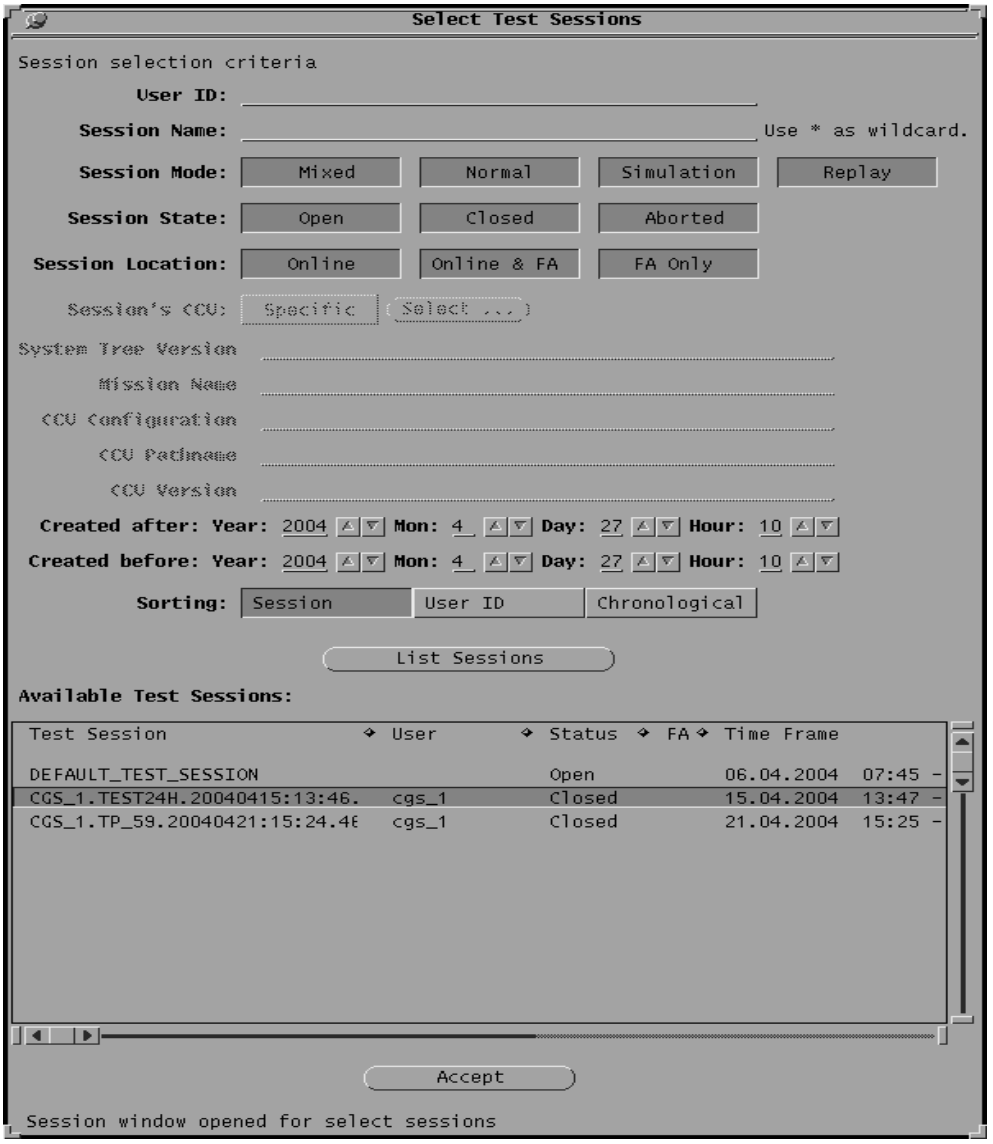


Figure 8–6 : *Select Test Sessions Window*

The user may select up to ten sessions in this list.

The user may display some more information about one session or delete a session, performing the following:

- Select one session in the list
- Click with the right button of the mouse

A menu appears proposing the following options:

- View ...

Get some more information about the session, like : System Topology table information , Test configuration used at the time of recording, Version id table used at the time of recording .....

- Session Size...

The user may see the sizes for each TRDB data types for the session : Archive files, Events, Engineering Values Logbook (EVL) and configuration file.

Copyright per DIN 34

- *Knowing the sizes of a session may be important before exporting or archiving on the MO device. The user shall know that the events table are translated into unix files before exporting or archiving. The number of events is then relevant to calculate the needed space and not the size of the Oracle tables.*
- *Considering the size of the archive files and EVL , the user can control that data has been generated for this session. Without such data files, TEV can not generate Data Sets.*
  - Delete....
- *Restrictions on Deletion :*
  – The user must have the privilege to delete a test execution session.
  – Sessions cannot be deleted if they are allocated to another user (another TEV user has already selected these sessions)
  – Sessions cannot be deleted if TEV tools are running
  – the DEFAULT_TEST_SESSION cannot be deleted through TEV

Once the user has selected sessions in this list, a click on the button **Accept** will accept these sessions as selected, the window **TEV : Select Test Sessions** disappears and the selected sessions appear in the list **Selected Execution Sessions** in the window **TEV:Select Execution Sessions.**

8.7.2.2 The Different Functionalities to Select a CCU

In some cases, the user shall select explicitely a CCU :

- when the default selected CCU (the one from the newest session) do no fit the actual needs
- when the CCU to be selected is one from the default test session
- when only a CCU, and no sessions, shall be selected

To select the CCU from a particular session, the user clicks on the button **Select CCU from Selected Execution Sessions ....**

Copyright per DIN 34

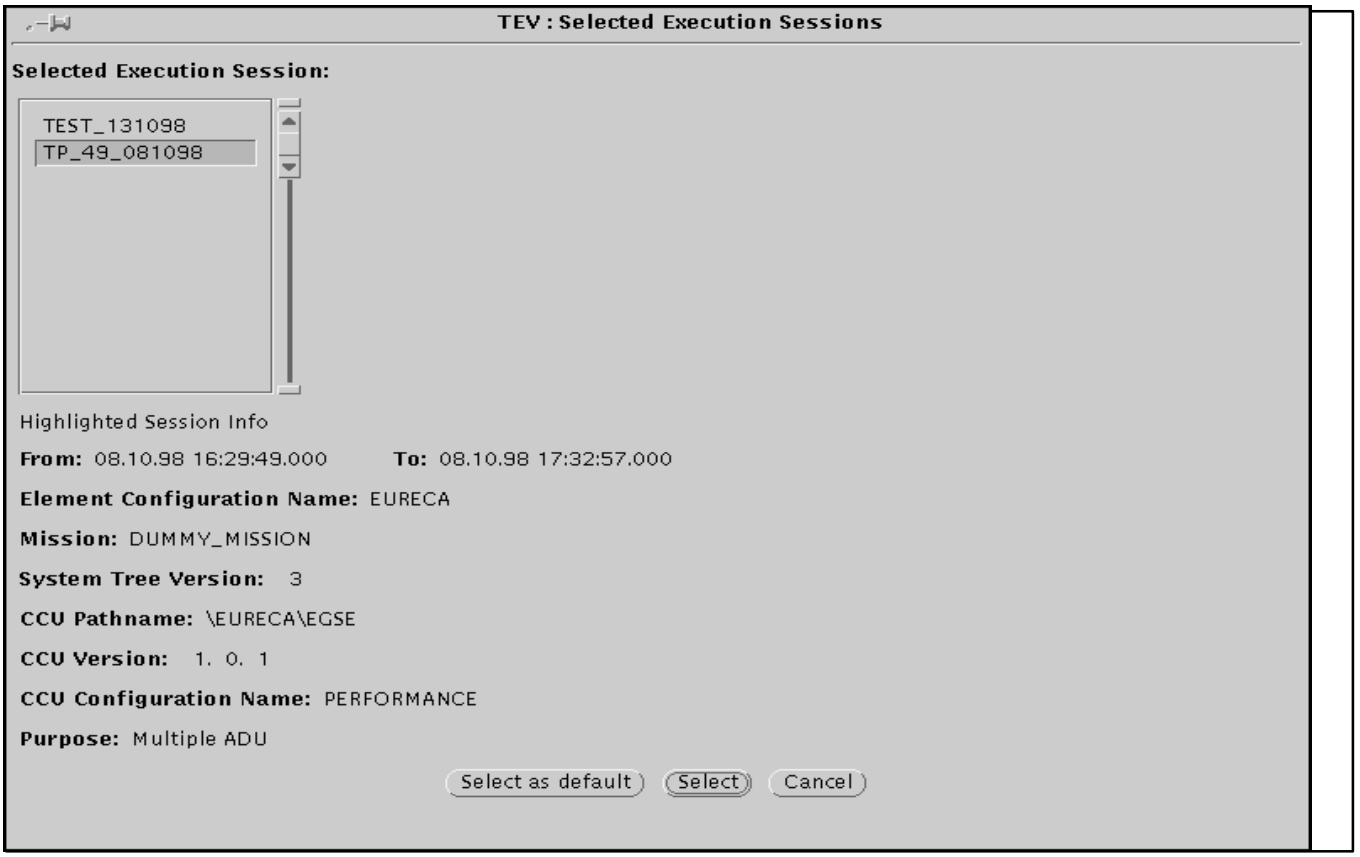


Figure 8–7 : *TEV : Selected Execution Sessions Window*

Select a CCU from a Selected Session

- Click on the button **Select CCU from Selected Execution Sessions**. The window **TEV : Selected Execution Sessions** appears. (Figure 8–7)
- Click on the session in the list for which the CCU information shall be displayed.
- Click on the **Store as Default** button. This CCU will be always selected by default if no session are selected. This information will be kept over after quitting TEV, for the same Unix user.
- *The Default CCU is the one which is selected, even if no sessions are selected.*
  *This CCU will also appear as default in the window **TEV:CCU List in MDB.** The Default CCU information is kept over after quitting TEV for the same Unix user*
- *The text fields below the Highlighted Session Info heading are read–only and contain the informations about the selected session in the list.*
- Click on the **Select** button to select the CCU
- *In the area **TEV: Select Execution Sessions : Seleted CCU** appears the CCU which is selected.*

Copyright per DIN 34

To select the CCU from the Mission Database, the user clicks on the button **Select CCU from MDB ....**

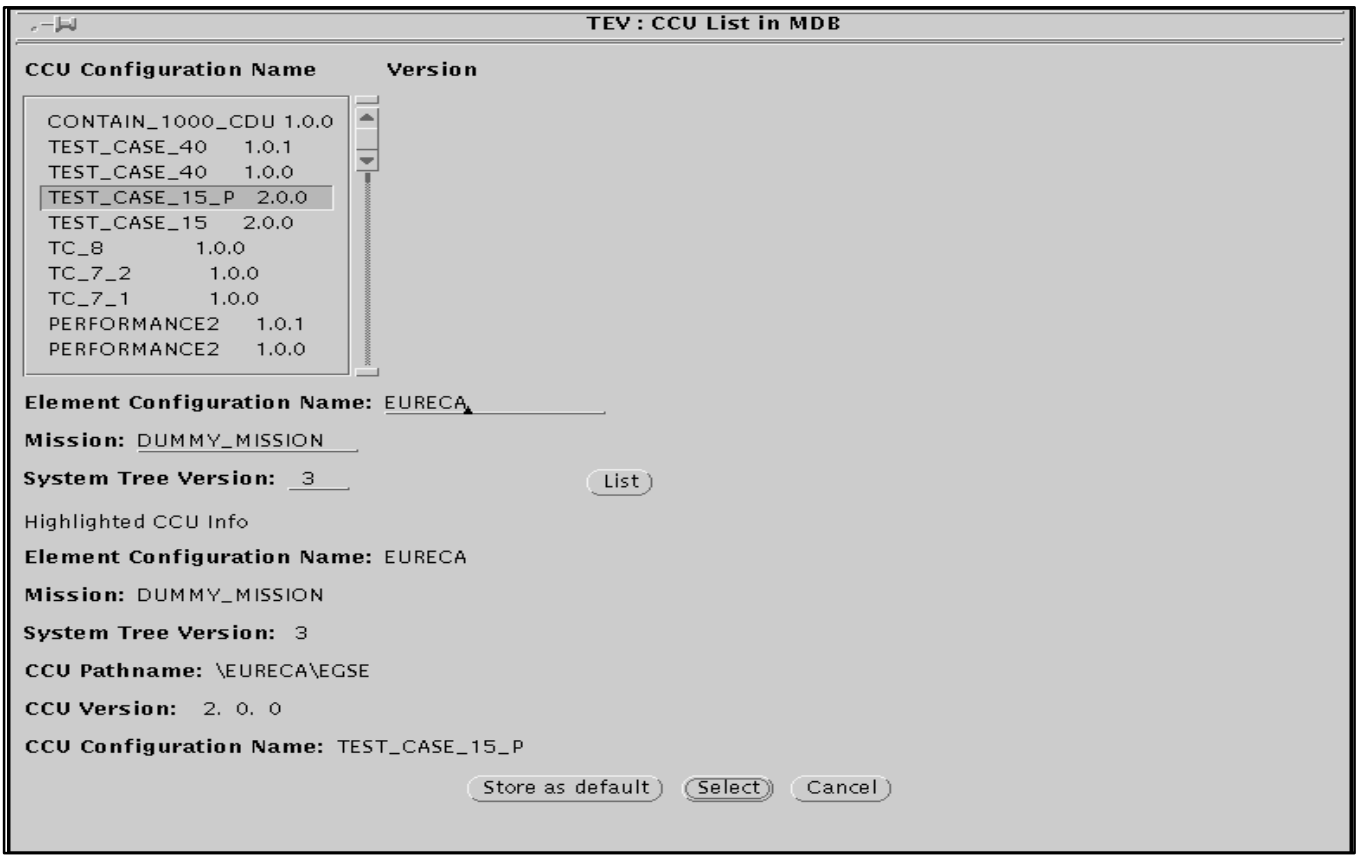


Figure 8–8 : *TEV : CCU List in MDB Window*

Select the CCU from the MDB

- Press the **Select CCU from MDB ...** button. The window **TEV: CCU list in MDB** appears.(Figure 8–8).
- *The initial CCU list is obtained for the element configuration, mission and system tree version from the default CCU. If no default CCU was previously stored, the list and fields are empty.*
- Enter the desired element configuration, mission and system tree version, then press the **List** button.
- The selected CCU can be stored as default : click on the **Store as Default** button**.**
- *The Default CCU is the one which is selected, even if no sessions are selected.*
  The Default CCU information is kept over after quitting TEV for the same Unix user
- Click on the **Select** button.
  The CCU is selected and appears in the area **Selected CCU** in the **TEV:Select Execution Sessions** window.

Copyright per DIN 34

To select the CCU from the Default Test Session, the user clicks on the button **Select CCU from Default Execution Sessions ....**

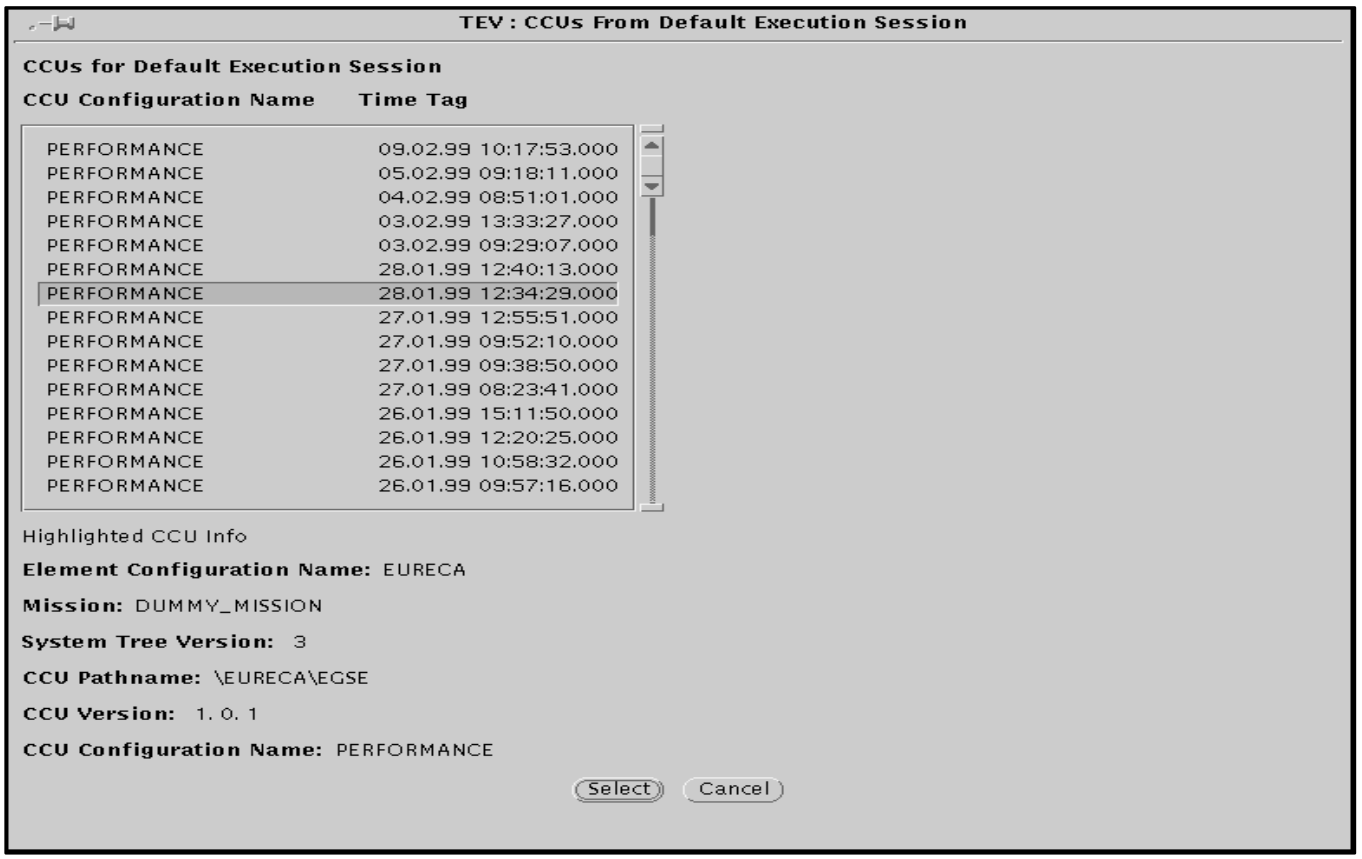


Figure 8–9 : *TEV: CCUs From Default Execution Session Window*

Select the CCU from the List of CCUs Used by the Default Test Session

- Press the **Select CCU from Default Execution Session...** button. The window **TEV: CCU list in MDB** appears.(Figure 8–9).
- Click in the list to select on CCU (and obtain the information about this CCU)
- Click on the **Select** button.
  The CCU is selected and appears in the area **Selected CCU** in the **TEV:Select Execution Sessions** window.

8.7.2.3 Validate the Selections :

Validate the Selection of Execution Sessions and CCU

- Press the **Accept** button on the bottom of the **TEV : Select Execution Sessions** window. If the selection is accepted by TEV, the window is iconised immediately.

A compliance check is performed to assure that the selected CCU exists in the configuration database. The selected sessions will be allocated on a successful accept, this prevents deletion or archiving by another

Copyright per DIN 34

user. The allocation is cleared whenever a re–initialisation is performed (i.e a new **Accept**) or when the user quit TEV.
An **Accept** cannot be performed if another TEV tool is open.

The Reset button reverts to the state of the last successful **Accept**.

- *An **Accept** will be performed even if the sessions selected are already selected and in use by another TEV user. This means that none of these users will be able to perform deletion or archiving for these sessions.*
- *The message :* **Warning : CCU with no proper consistency status selected from MDB** *don't prevent the user to evaluate the selected session(s), as far as those information needed from the MDB are consistent.*

8.7.3 The Export/Import Tool

Execution sessions and evaluation sessions may be exported through this tool. Export means to copy the selected data from a session to the FA medium. To import is to copy a named session from the FA medium to the TRDB.
The TRDB keeps no references to an exported session. An exported session is a duplicate of a session which stays unchanged in the TRDB. The disk where the exported session has been copied can be taken over and used within an other TRDB. It is possible to export and delete sessions in one step.
To access the Export–Import Tool, the user must select the 'Export/Import...' item in the *Sessions* menu (Figure 8–2).

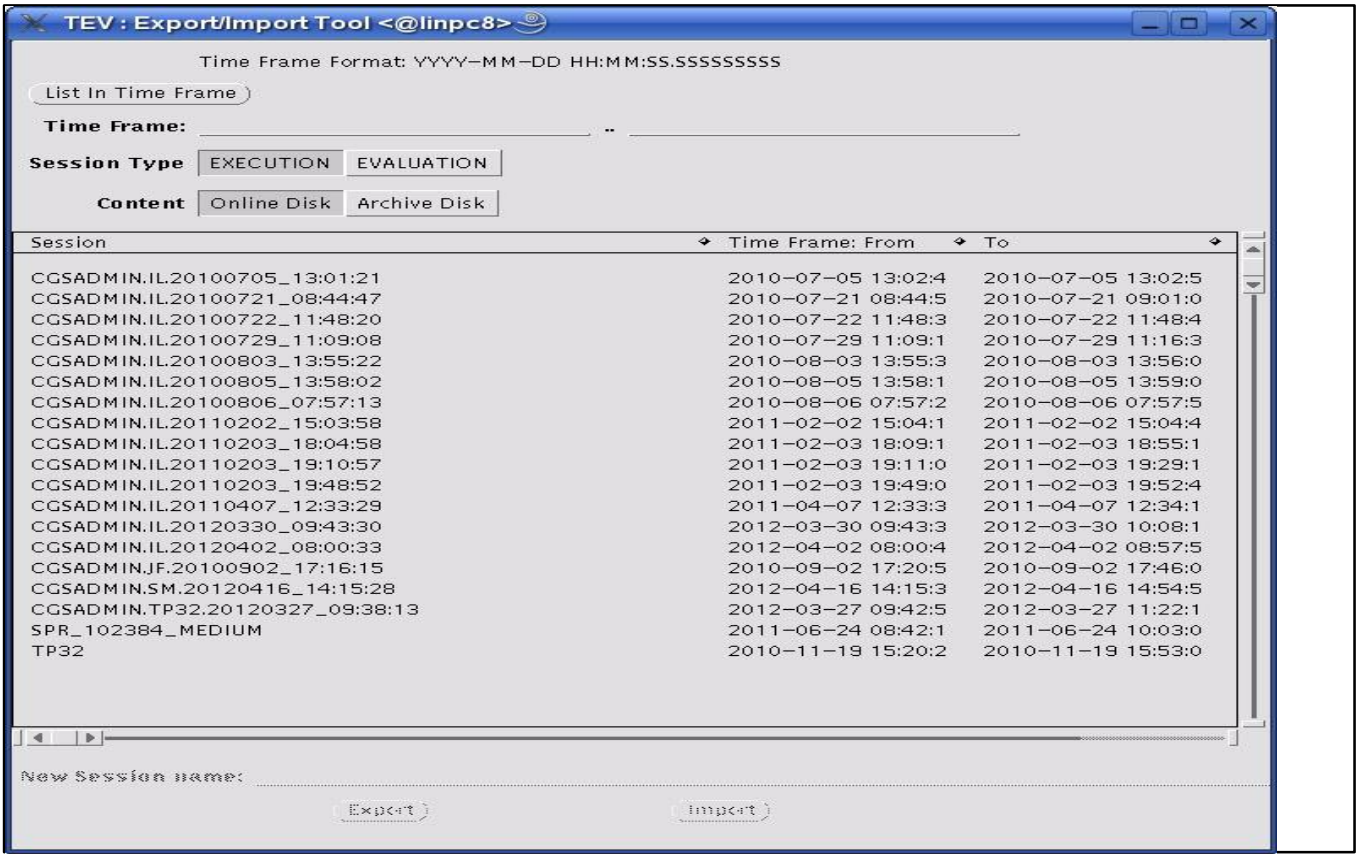


Figure 8–10 : *Export/Import Tool Window*

Copyright per DIN 34

The **Time Frame** field is initially set to the current date and time. The **List in Time Frame** button allows the user to list the On Line Sessions included into the time frame selected.

Export an Evaluation Session

- Select **Session Type : Evaluation**
- Select **Content : Online Disk**
- *The list of evaluation sessions included in the time frame is displayed.*
- Select an Evaluation Session by clicking on its name in the online session list.
- If you want to change the original name: Enter a new name in the field **New Session name**
- Press the **Export** button.

Export an Execution Session

- Select **Session Type : Execution**
- Select **Content : Online Disk**
- *The list of execution sessions included in the time frame is displayed.*
  Note that some of these sessions can be partially online.
- Select one ore several Execution Session(s) by clicking on its name in the online session list.
- If only one session is selected and if you want to change the original name: Enter a new name in the field **New Session name.**
- Press the **Export** button.
- An confirmation window pops up.
- If you want to delete the online session(s) after export: Enable the **Delete online session after export** check box
- Press the **Export** button.

Import a Session

- Select **Session Type : Execution** or **Evaluation**
- Select **Content : Archive Disk**
- Select a Session by clicking on its name in the archive session list.
- If you want to change the original name: Enter a new name in the field **New Session name.**
- Press the **Import** button.

Copyright per DIN 34

8.8 Data Evaluation Tools

8.8.1 Evaluation Tools Overview

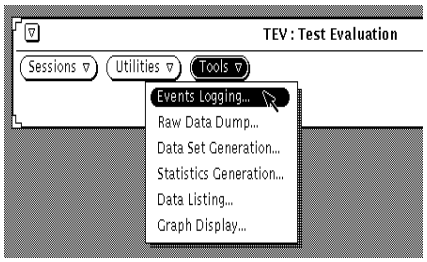


Figure 8–11 : *The Different Evaluation Tools (note: in CGS 6.3.2, the menu items Statistics Generation... and Graph Display... are available only in backward compatibility mode)*

Each evaluation tool allows to extract specific data from the TRDB and format them in a specific way.
The Events Logging Tool allows to extract events and displays them in a listing.
The Raw Data Dump Tool allows to extract ADUs and GDUs packets from archive files and displays them as raw values.
The Data Set Generation Tool is an intermediate tool which does not provide a result directly accessible to the user : it allows to extract measurements values from archive files or from engineering values logbooks and generate a file of engineering values, called a Data Set.
The Statistics Tool provides statistics on a set of values stored into a Data Set.
The Listing Tool provides the listing of a set of values stored into a Data Set.
The Graphics Tool provides graphics from values stored into a Data Set.

Each tool saves its specific result files in an appropriate subdirectory under the tev_working_directory : **user_home/.cgs/tev** (also represented by the environment variable $VICOS_TEV_WD). The user can access these results directly in these subdirectories.

Subdirectory for :

The Events Logging Tool result files : $VICOS_TEV_WD/RESULTS/EVENT_LIST
The Raw Data Dump Tool result files : $VICOS_TEV_WD/RESULTS/RAW_DATA_DUMP
The Data Set Generation Tool result files : $VICOS_TEV_WD/RESULTS/DATA_SET
The Statistics Tool result files : $VICOS_TEV_WD/RESULTS/STATISTICS
The Listing Tool result files : $VICOS_TEV_WD/RESULTS/DATA_LIST
The Graphics Tool result files : $VICOS_TEV_WD/RESULTS/GRAPHS

Up to four instances of each evaluation tool may run simultaneously on the screen. Each tool main window title bar contains the identification name of the tool, the instance number of the tool and the name of the current definition (if not null). Each tool main window contains a window Footer displaying information on the last operation. Most of the tools main window contains the menus *Definition* and *Result*, which are explained below.

8.8.2 The *Definition* Menu

The main window of every evaluation tool is based upon the concept of current definition.
A definition defines, for each type of evaluation tool, criteria of selection for data and parameters for execut-

Copyright per DIN 34

ing the evaluation. The user shall either used the NULL definition which appears by default when opening a tool or defines their own definition. The current definition is used for generation of an evaluation result. The current definition is the definition actually displayed in the main window.

The *Definition* menu provides the applicable operations on definitions : *Load, Save, Delete, Clear* and *Reset*, as shown in the next figure.

Operations in the menu ending with an ellipsis (...) need a definition name (name of the file containing the definition). When these kind of operations are started, a pop–up window with a list of existing definitions (for the current tool type) is displayed for the user to select a name or enter a name directly via the keyboard.

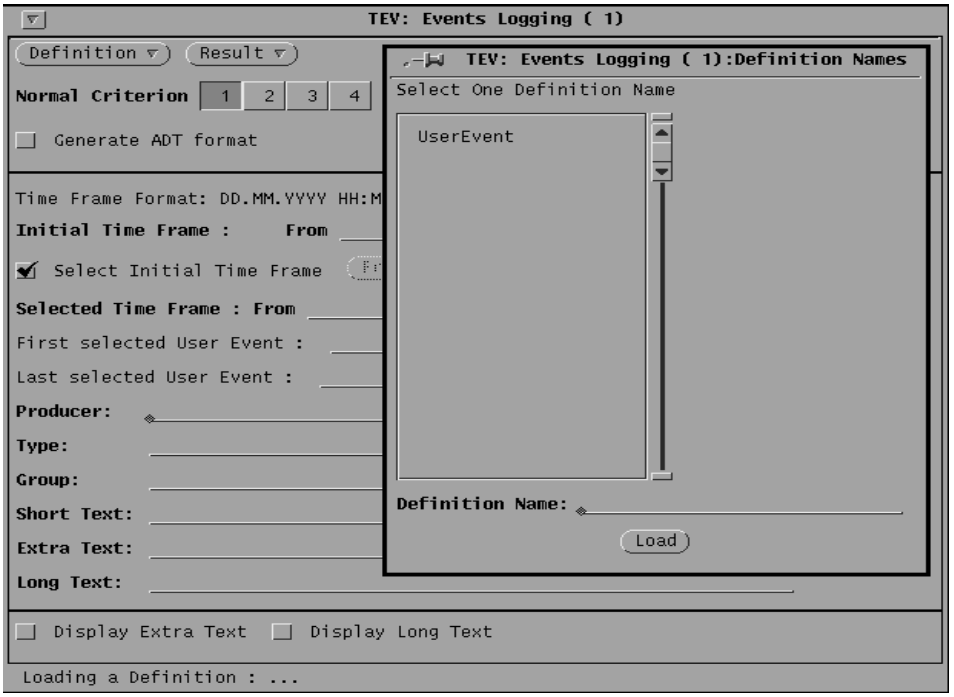


Figure 8–12 : *Definition Menu and its Pop–Up Menu*

The *Load...* operation loads the named definition, which becomes the new current definition. The definition fields in the current evaluation tool window are overwritten, without warning. The name of the definition is displayed in the title bar. The definition file is loaded from the directory tev_working_directory*/DEFINITIONS/*result_type_of_the_tool.

If the titles from two selected user events where stored in the definition, these appear in the fields : **First Selected User Event** and **Last Selected User Event**. In this case the **Selected Time Frame** fields become empty. The selected Time Frame is calculated by TEV when the user starts the **Exec&Display** process. The calculated Selected Time Frame can be different for every session containing the same User Event titles.

Copyright per DIN 34

Select a Predefined Definition

- Select **Definition –> Load...** from the pop–up menu.
- Select the name of the desired definition from the **Definition Names** list
- Press the **Load** button.

The *Save...* operation saves all the fields of the definition into a file.
The titles of the two selected user events in the fields : **First Selected User Event** and **Last Selected User Event** are part of the definition and will be saved when the definition is saved. This allows the user to work with different sessions, using the same definition of a Time Frame described through User Events. These fields are saved prior to the Selected Time Frame.
The definition is first checked for errors and saved under the given name if there is no error. If the chosen name already exists, a notice box asking for confirmation of overwriting is displayed with two buttons 'Overwrite' and 'Cancel'. In case of successful completion, the name of the definition in the tool window title is modified to the new name. The saved definition still stays the current definition. The definition file is saved into the directory of the user under tev_working_directory*/DEFINITIONS/*result_type_of_the_tool.

Save a Definition

- Tailor the definition for users needs.
- Select **Definition –> Save...** in the pop–up menu.
- Type the name in the **Definition Name ____** field.
- Press the **Save** button.

The *Delete* operation allows to delete a selected definition.

The *Clear* operation resets the current definition to the null definition. The fields are reset to default values as if the tool had just started. The definition name in the title is also reset. There is no warning given.

The *Reset* operation changes the current definition to revert to the state of the last successful *Load* or *Save* operation. It can be used to come back to a correct state after user modifications have not been accepted.

8.8.3 The *Result* Menu

This menu offers the operations on Evaluation Result Files :

The *Load...* operation reads a result file and displays it into a specific window. The title of this window contains the tool reference, its instance number and the result file name. The result file can be loaded either from the directory tev_working_directory*RESULTS/*result_type_of_the_tool (*source:Working Directory*), or from the current open Evaluation Session (*source:Test Result Data Base*).

The *Save...* operation saves the current result into the directory user_home*/wd/tev/RESULTS/*result_type_of_the_tool. (To save a result into the current open Evaluation Session, use the *Store* operation from the File Manager tool. See Chapter 8.5.1).

The *Exec&Display...* operation creates a result for the current definition. The definition is first checked for error. In case of error no execution takes place. During execution, further operations in the window are dis-

Copyright per DIN 34

abled and this is shown by the shaded title–bar and stopwatch pointer. However, operations in other windows are permitted.
If the window has been unpinned and the definition has not been changed since the last execution, invoking the *Exec & Display...* operation will not generate the result again but simply display the previously generated result.
When the execution is complete, the result is displayed in a separate result window. The title of this window contains the tool reference and its instance number. The user can browse at the result in the usual OpenWindows way.

Exec & Display a Definition

- Select **Definition –> Load...** from the pop–up menu.
- Click on the definition name in the list.
- Press the **Load** Button.
- Select **Result –> Exec & Display** from pop–up menu.

8.8.4 Select a Time Frame

The **Select Initial Time Frame** check box is enabled by default. In this case all the session(s) data are selected. When this box is enabled the **Selected Time Frame** fields cannot be modified. To change the time frame the user must disable the check box first.

There are than 3 alternatives to specify a time frame :

- Enter it directly in the *Selected Time Frame* fields,
- Copy it from the time frame of an Execution Session,
- Copy it from the time tags corresponding to 2 User Events.

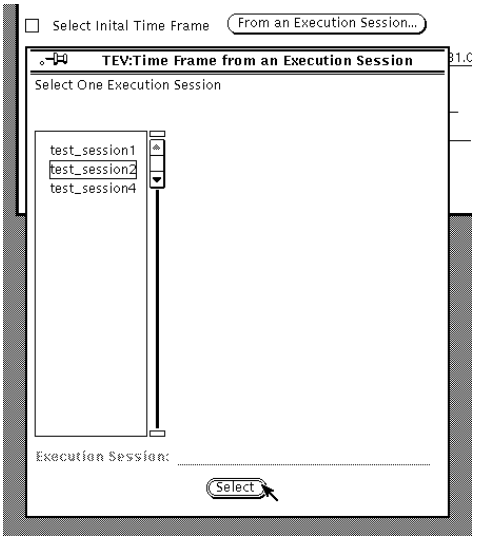


Figure 8–13 : *Selecting a Time Frame from an Execution Session*

Copyright per DIN 34

Enter directly a Time Frame.

- Disable the **Select Initial Time Frame** check box by clicking in the check box.
- *The* Selected Time Frame *fields become available. The* Selected Time Frame: From *and* To *fields are shown underlined.*
- Enter a time frame into the **Selected Time Frame** fields.
- *The day (DD), month (MM), year (YY), hour (HH), minutes (MM) and seconds (SS) fields must be two–digit numbers. The milliseconds field (MSS) has three digits. Note that* HH, MM, SS *and* MSS *have default value of 0 and are optional from the right to the left.*

Select a Time Frame from an Execution Session

- Disable the **Select Initial Time Frame** check box by clicking in the check box.
- Press the **From an Execution Session..** button.
- *The window* TEV: Time Frame from an execution session *appears, containing the list of selected Execution Sessions.*
- Select one execution session
- Press the **Select** button. The time frame of the session will be copied to the **Selected Time Frame** fields.
- *TEV uses the text entered in the fields* ***First Selected User Event*** *and* ***Last Selected User Event*** *prior to the* ***Selected Time Frame : remove the text in the fields First Selected User Event*** *and* ***Last Selected User Event to force TEV considering the Selected Time Frame inputs.***

Copyright per DIN 34

Select a Time Frame from two User Events

- Disable the **Select Initial Time Frame** check box by clicking in the check box.
- Press the **From User Events...** button.

The window TEV: Time Frame from User Events *appears, containing the User Events existing in the currently selected Execution Sessions.*

- Select the Time Type using the **Time Type** boxes.
- Click on two user events.

The two time tags will be ordered and a time frame constructed.

- Press the **Select** button. The new time frame will be copied to the **Selected Time Frame** fields.

The titles of the two selected user events appear in the corresponding fields : ***First Selected User Event*** *and* ***Last Selected User Event****.*

- OR :
- Enter directly the User Events titles in the corresponding fields : ***First Selected User Event*** *and* ***Last Selected User Event****.*

The text entered in the fields ***First Selected User Event*** *and* ***Last Selected User Event*** *are part of the definition and will be saved when the definition is saved. This allows the user to work with different sessions, using the same definition of a Time Frame described through User Events. TEV uses the text entered in the fields* ***First Selected User Event*** *and* ***Last Selected User Event*** *prior to the* ***Selected Time Frame : remove the text in the fields First Selected User Event*** *and* ***Last Selected User Event to force TEV considering the Selected Time Frame inputs.***

To enable the Initial Time Frame *again, enable the* Select Initial Time Frame *check box.*
The Selected Time Frame *fields cannot be modified and hold a copy of the* Initial Time Frame

Copyright per DIN 34

8.8.5 Events Logging Tool

To access *Events Logging* tool operations, the user must select the **Events Logging...** item in the ***Tools*** menu.

The *Events Logging* Tool window (Figure 9–18) appears:

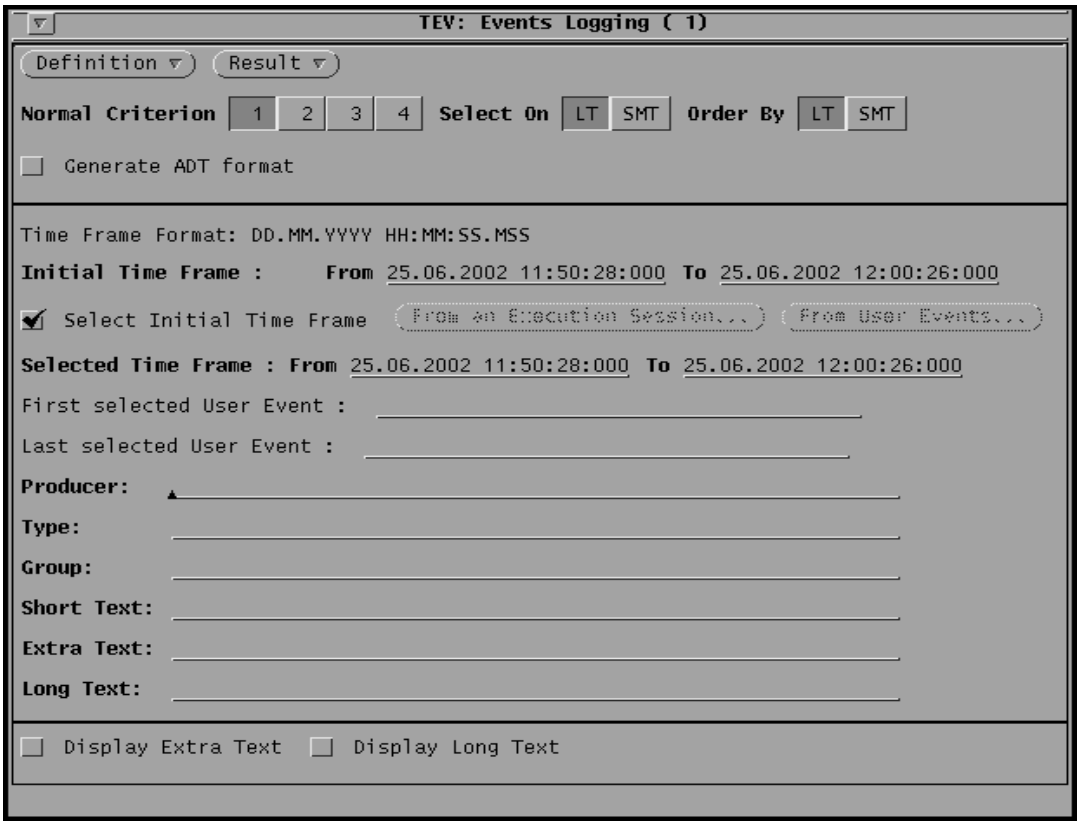


Figure 8–14 : *The Events Logging Tool Main Window*

The **TEV: Events Logging** Tool main window shows the two menu buttons labelled **Definition** and **Result** and a panel with the contents of the current definition.

For more information about *Definition* and *Result* operations refer to the Chapters 8.4.2 and 8.4.3.

> ***It is necessary to generate a events listing result in ADT format if this result shall be merged with one DataSet (see Utilities:Merge Dat Sets/Events).***

8.8.5.1 Create a Definition

A Events Logging Tool definition allows the user to select events and to define the output format of events.

The user can either create a new definition or select and modify an already existing definition.

> *Note that it is not mandatory to give any selection criterion in the Events Logging Tool. The user can use the default definition without changing the entries in the **Events Logging** main window. In this case **all** events which happened during the Initial Time Frame and for the selected sessions will be selected from the TRDB and put into the result.*

Copyright per DIN 34

Levels of Selection Criteria :

There are three levels of criteria in the **Events Logging Tool**:

- An ***Atomic*** criterion is the simplest selection mechanism and allows to specify the value of a single field of a logged event. The ***Sender*** field allows to include only events which originate from a given application. The ***Class*** and ***Location*** fields allow to extract only events of specific classes (INFO, WARNING, ERROR, FATAL, DEBUG, ...) and locations (LOG, UEVT, HLCL, \.<AP_NAME>, DACQ, DGEN, CDB, TES, CIS, ...). The **Title** and **Text** fields specifiy a string which must be searched for in the text associated to events; only events containing the substring in their text fields will be selected. The ***Selected Time Frame*** is also an atomic criterion used to restrict the event selection in time.
  If the text field of an ***Atomic*** criterion is left empty, all the events will match the criterion with regard to this field .
  In the ***Title*** *resp.* ***Text*** fields, combination of + (and) , – (and not) and | (or) can be used.
  In the **Sender**, **Class** and **Location** fields the use of + combination makes no sense, since it is not possible to select an event having several senders, classes or locations (null result will occur). For example :
  - ***Sender*** ***:*** *TES_01* **|** *HCI_01* : select all the events produced by TES_01 **or** by HCI_01. *TES_01* **+** *HCI_01* would mean having the Producer TES_01 and HCI_01.
  - ***Class*** ***:*** *– INFO* means select all the events which do **not** have the type INFO.
  - ***Location*** ***:*** *HLCL* means select the events belonging to the location HLCL
  - ***Title/Text :*** *+ import – VALUES* : select all the events containing in the title resp. text the string “import” **and not** the string “VALUES”. Wildcards (like *) can not be used. *+ import | – VALUES :* select all the events either containing the string *import* or do not containing the string *VALUES*
- A ***Normal*** criterion is the combination of the different types of atomic criteria. The parts are joined by ”AND” relationships.
  (e.g. (Time between t1 and t2) AND (Sender = TES_01 or HCI_01) AND (Class = not INFO) AND (Location unspecified) AND (Title includes ’import’ but not ’VALUES’)).

Copyright per DIN 34

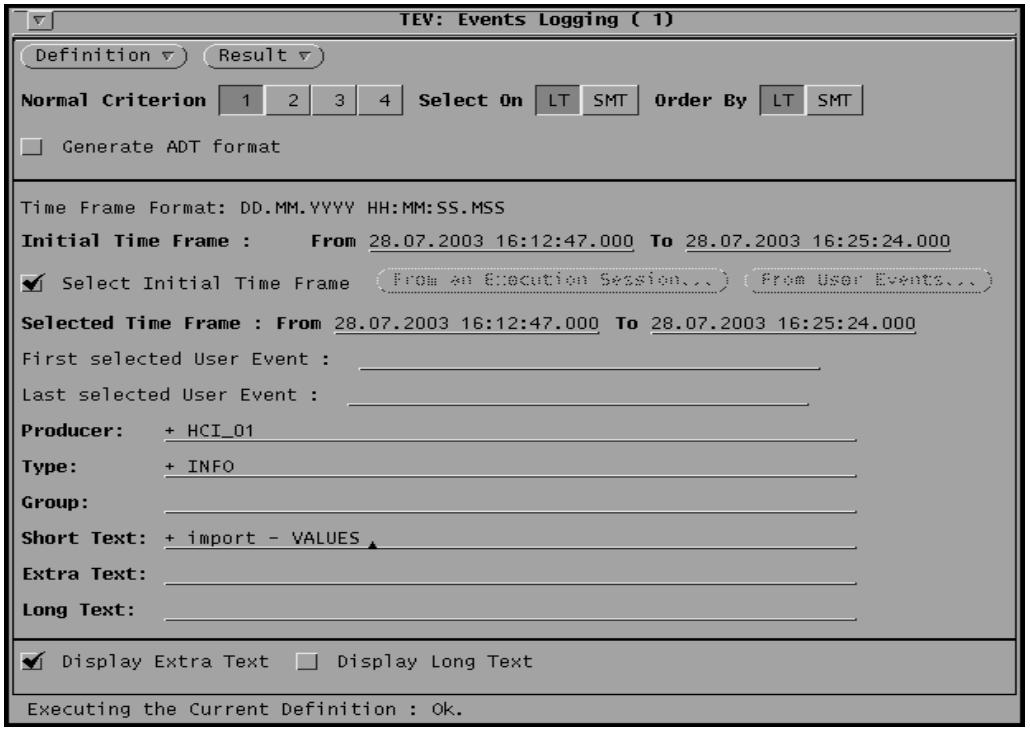

TEV: Events Logging (1)
Definition
Result
Normal Criterion 1 2 3 4 Select On LT SMT Order By LT SMT
Generate ADT format
Time Frame Format: DD.MM.YYYY HH:MM:SS.MSS
Initial Time Frame : From 28.07.2003 16:12:47.000 To 28.07.2003 16:25:24.000
Select Initial Time Frame
From an Execution Session...
From User Events...
Selected Time Frame : From 28.07.2003 16:12:47.000 To 28.07.2003 16:25:24.000
First selected User Event :
Last selected User Event :
Producer: + HCI_01
Type: + INFO
Group:
Short Text: + import - VALUES
Extra Text:
Long Text:
Display Extra Text
Display Long Text
Executing the Current Definition : Ok.

Figure 8–15 : *Event Logging Window*

Use of a Specific Normal Criterion

- Select among the **Normal Criterion** boxes the number 1,2,3 or 4
- *The criteria fields appear empty, except for the time frame, set by default to the initial time frame. If these criteria fields are not modified, the normal criterion is considered as empty.*
- *The Normal Criterion 1 is used by default, the others Normal criterion are by default empty.*

- A ***Combined*** criterion is a combination of up to 4 *Normal* criterion. The Normal criterion are joined by ”OR” relationships. An empty *Normal* criterion is ignored. To be taken into account into the combination, a Normal criterion just has to be defined (not empty).
  Example : if Normal Criterion 1,2 and 4 are not empty, events satisfying (Normal Criterion 1) OR (Normal Criterion 2) OR (Normal Criterion 4), are selected.

Copyright per DIN 34

General Procedure to Specify the Definition :

Specify an Events Logging Definition

- Select the time base LT (local time) or SMT (simulated mission time) for **Select On** and **Order By** time types.
- Select a **Format** TXT, CSV or ADT (binary)
- Select a **Normal Criterion** 1,2,3 or 4
  - Select a Time Frame.
  - Type some name(s) of event sources (separated with the or symbol |) in the **Sender** field.
  - Type event classes in the **Class** field. (separated with the **or** symbol |)
  - Type the locations in the **Location** field. (separated with the **or** symbol |)
  - Type strings in the **Title** resp. **Text** field. (separated with the symbols + , – or |)
- Define a set of criteria for each of the set 1,2,3 and 4
- Enable the **Display Text** check box. The texts of the events will be included in the result.

8.8.5.2 Generate a Result

By clicking on *Exec&Display* button in the *Result* menu, the tool will select the logging data from the Test Result Database according to the current definition.

The generation of the result (performed by the DBS tool) occurs in two steps:
1. extraction of the events from the Oracle Database through an Oracle Request
2. formatting of the events as TXT or CSV if one of these formats is selected

The result is generated in a temporary file in the directory ***$CGS_HOME/local/tmp/tev/*tmp_machine_user**.
A window proposes the possibility to interrupt the formatting of the events. The Oracle Request can not be interrupted: the user interruption will only be taken into account after completion of the Oracle Request.

- *Interrupt the Events Listing generation when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. In this case the user will get the result until the interruption.*
- *The Events Listing generation may be interrupted automatically because of a "Disc Space problem". In this case a result is generated until the interruption. The user shall first recover space on the disc before saving the result by selecting **"Result:Save".** Afterwards the user may recover some more place by deleting the temporary files in **$CGS_HOME/local/tmp/tev/tmp_machine_user**. The result can be then re–loaded normally by selecting **"Result:Load"** ...*
  *To avoid space problem when a big amount of events has been stored for the session, it may help first not selecting **"Display Text"** and consider which time period is of interest.*
  Running TEV on the DBS–Server decreases considerably the processing time.

To **save** the result use the menu option **Result:Save.....** This operation performs a copy from *$CGS_HOME/local/tmp/tev/*tmp_machine_user*/current_temporary_events_listing* in the directory tev_working_directory*/RESULTS/EVENT_LIST.*

Copyright per DIN 34

To **save** the result into the TRDB (i.e into an Evaluation Session) use the **store** operation from the File Manager tool (see Chapter 8.9.1 File Handling).

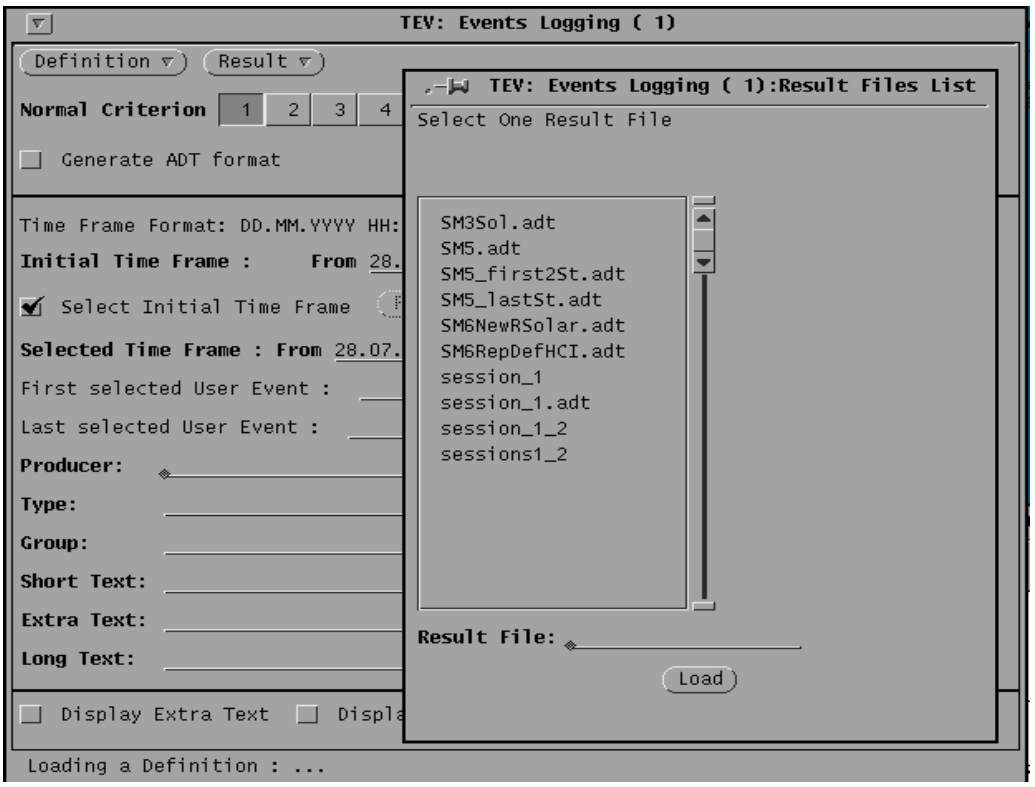


Figure 8–16 : *Result Menu and the List of Result Files*

8.8.5.3 Generate an ADT Result

Instead of generating the result in an TXT or CSV file the user can generate the result in an ADT format. The ADT format are ADA structures which are readeable througth ADT procedures. These ADT procedures are provided by the TEV_API library. Using this format allows the user to work with Events Listings within external ADA programs.

Copyright per DIN 34

Generate an ADT Result

- specify a definition (see chapter 8.4.5.1)
- select the **Format ADT**
- select **Result:Save...**
- Enter a filename for the ADT result and click on **Save** in the **TEV: Events Logging (i) : Result File Names** window

The result is directly stored in the directory **tev_working_directory/*RESULTS/EVENT_LIST***. The extension **.adt** is automatically added to the filename.*

*The ADT result can be displayed in TXT format by loading it: use **Result:Load... ,** select the ADT filename and click on **Load**.*

It is necessary to generate a events listing result in ADT format if this result shall be merged with one DataSet (see Utilities:Merge Dat Sets/Events).

Copyright per DIN 34

8.8.6 Raw Data Dump Tool

The Archive (i.e. a set of raw data files) contains two data formats:

- incoming data is received and stored as ADU (**A**cquisition **D**ata **U**nits)
- outgoing data is stored as GDU (**G**eneration **D**ata **U**nits)

ADUs and GDUs are stored in the Archive during the online operations and may be dumped by the Raw Data Tool after or during the test.

To access *Raw Data Dump* tool operations, the user must select the **Raw Data Dump...** item in the ***Tools*** menu.

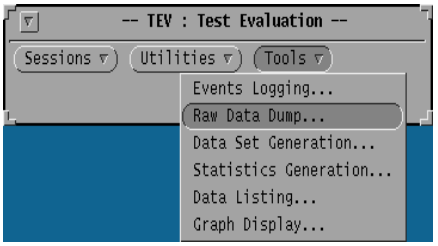


Figure 8–17 : *Start of the Raw Data Dump Tool*

The **Raw Data Dump** window (Figure 8–18) appears:

Copyright per DIN 34

TEV: Raw Data Dump (1)
Definition
Result
Edit Format
Format: TXT ADT
Select On: Undefined Arch SMT Arch LT SMT LT PT Alt: Undefined Arch SMT Arch LT SMT LT PT
Order By: Undefined Arch SMT Arch LT SMT LT PT Alt: Undefined Arch SMT Arch LT SMT LT PT
Reject Duplicates
Time Frame Format: DD.MM.YYYY HH:MM:SS.MSS
Session Time Frame : From 08.10.2007 16:24:21.000 To 08.10.2007 16:46:08.000
Select Initial Time Frame
From an Execution Session...
From User Events...
Selected Time Frame : From To
First selected User Event :
Last selected User Event :
TES Node: TES_01
Data Units: ADUs GDUs ADUs + GDUs
Select ADUs And GDUs From Mission Database:
Pathname:
Select ADUs...
Select All ADUs
Select GDUs...
Select All GDUs
Select ADUs And GDUs From Archive:
Select ADUs...
Select GDUs...
No Of ADUs Selected: 5
No Of GDUs Selected: 3
Nicknames Pathnames
Q29_CCSDS1_B \EURECA\EGSE\TEST\29_TEV\TN_1\A
Q29_CCSDS1_A \EURECA\EGSE\TEST\29_TEV\TN_1\A
Q29_STRUC1_A \EURECA\EGSE\TEST\29_TEV\TN_1\A
Q29_STRUC1_B \EURECA\EGSE\TEST\29_TEV\TN_1\A
Q29_UNSTR1_A \EURECA\EGSE\TEST\29_TEV\TN_1\A
Nicknames Pathnames
Q29_CCSDS_TC \EURECA\EGSE\TEST\29_TEV\TN
Q29_ANA_STIMULUS \EURECA\EGSE\TEST\29_TEV\TN
Q29_DISC_STIMULUS \EURECA\EGSE\TEST\29_TEV\TN
Selecting Items : Ok.

Figure 8–18 : *Raw Data Dump Window*

The Raw Data Dump tool has no **Exec&Display** functionality. Two different functionalities are provided instead to the user:

Copyright per DIN 34

- ***Result:Start Packets Navigator*** : to consult directly the packets dumped in the Archive (no result files are generated)
- ***Result: Save Dump In File*** : extract packets and save them in a result file in TXT (ASCII text) or ADT (binary) format

*To re–use (**Result:Load...**) old ADT result files (CGS V4.xx), rename them first with the extension .adt.*

The Packets Navigator (or up to 4 Packets Navigators) may be use for a first quick analyse of the data. Reduce the Time Frame before starting the Packets Navigator(s) in order to optimize your investigation.

Then use the command ***Result: Save Dump in File*** to generate a result file containing the dumped packets of interest.

Both options use the Definition that the user has defined through the User Interface. The Definition can be empty. In this case all ADUs and GDUs packets which are dumped in the Archive Files are searched.

8.8.6.1 Create a Definition

Copyright per DIN 34

Create a Definition

- Select a Time Frame to limit the data to be dumped or displayed.
- *The selection of a Time Frame is identical in each Evaluation Tool. Refer to "Select The Time Frame" paragraph in the Chapter 8.4.4*
- Select the time base for data selection (wrt. time frame specified) using the **Select On:** choice**.**
  - Undefined — no selection with respect to the selected time frame is applied (selected time frame is ignored), all data is included
  - SMT — select on (simulated) mission time as found in ADU/GDU (AT_TIME of GDU, if AT_TIME base is SMT)
  - LT — select on local time as found in ADU/GDU (AT_TIME of GDU, if AT_TIME base is LT)
  - Arch LT — select on local time when packet was received and archived by TES
  - Arch SMT — select on (simulated) mission time when packet was received and archived by TES
  - PT — select on packet time (from secondary header of CCSDS packet, i.e. packet generation time (TM packet) resp. onboard execution time (TC packet)
- If data is selected on PT, select the alternative time base for selection of non–CCSDS packet data using the **Alt Select On:** choice.
  - Undefined — see above
  - SMT — see above
  - LT — see above
  - Arch LT — see above
  - Arch SMT — see above
  - PT — all non–CCSDS raw data packets are discarded since there is no packet time, thus this filter option allows to reject all non–CCSDS packet data
- Select the time base for data ordering using the **Order By:** choice.
  - Undefined — no specific sorting is applied, data is ordered as found in the Archive
  - SMT — order by (simulated) mission time as found in ADU/GDU (AT_TIME of GDU, if AT_TIME base is SMT)
  - LT — order by local time as found in ADU/GDU (AT_TIME of GDU, if AT_TIME base is LT)
  - Arch LT — order by local time when packet was received and archived by TES
  - Arch SMT — order by (simulated) mission time when packet was received and archived by TES
  - PT — order by packet time (from secondary header of CCSDS packet, i.e. packet generation time (TM packet) resp. onboard execution time (TC packet)

Copyright per DIN 34

Create a Definition (continued)

- If data is ordered by PT, select the alternative time base to sort into the non–CCSDS packet data using the **Alt Order By:** choice.
  - Undefined see above
  - SMT see above
  - LT see above
  - Arch LT see above
  - Arch SMT see above
  - PT all non–CCSDS raw data packets are discarded since there is no packet time, thus this filter option allows to reject all non–CCSDS packet data
- Reject duplicate CCSDS packets using the **Reject Duplicates:** option. If enabled, duplicate CCSDS packets are ignored, i.e. multiple ADUs/GDUs with the same SID, the same time tag corresponding to the current ”Order By” option and the same sequence count (from primary header of CCSDS packet); note: to reject duplicates, usually Order By should be set to PT.
- Enter the name of the TES node having archived the raw data to extract in the **TES Node:** text field. By default, if no TES node is specified, raw data packets from all TES nodes involved in the selected execution sessions are included.
- Select the type(s) of data to extract using the **Data Units:** choice.
  - ADUs : include ADUs only
  - GDUs : include GDUs only
  - ADUs + GDUs : include ADUs and GDUs
- Select specific end items to be dumped or displayed (note that by default, if no speciific ADU or GDU is selected, all ADUs and GDUs are considered as being selected). To do so there are two possibilities (in either case, make sure that the appropriate type(s) of data is/are selected):
- 1) Select the ADUs/ GDUs from the Mission Database (Figure 8–19). This choice makes sense when the user knows exactly the pathname of the items to be selected, or when the user only wants to create a definition and has selected no session (and consequently no archive files).
- Fill the **Pathname:** text field with a virtual node name, which will be the root for the ADUs and GDUs pathnames to be selected.
  - **Select All ADUs** select all the ADUs existing under the pathname selected.
  - **Select All GDUs** select all the GDUs existing under the pathname selected.
  - **Select ADUs...** open a window to choose some ADUs among the list of all ADUs existing under the pathname selected.
  - **Select GDUs...** open a window to choose some GDUs among the list of all GDUs existing under the pathname selected.

Copyright per DIN 34

Create a Definition (continued)

*The user can get quickly the pathnames of the dumped items in the Archive as follows: Click on **Edit Format** and choose **Summary Format.** Quit the menu by clicking on **Accept.** Start **Result : Start Packets Navigator** and eventually interrupt the dump : The list of the dumped items (with time tags and without contents) appears. From this list the user can copy/paste the relevant pathname(s).*

OR

2) Select the ADUs/ GDUs from the Archive. (Figure 8–20).

- Press the **Select ADUs...** button.
- The list of the ADUs archived appears. In this list select the ADUs to be dumped.
- Press the **Select GDUs...** button.
- The list of the GDUs archived appears. Select in this list the GDUs to be dumped.

Choose the output format for the packets (see Chapter **8.4.6.2**)

Figure 8–19 : *Select the ADUs and GDUs from the Mission Database*

Figure 8–20 : *Select the ADUs and GDUs from the Archive*

Copyright per DIN 34

8.8.6.2 Consult the Packets Dumped in the Archive

Consult the Packets Dumped in the Archive

- Create a Definition. The definition can be empty, in this case all the ADUs and GDUs are selected.
- Select **Result : Start Packets Navigator** from the pop–up menu.

*The navigation tool appears. No result file is generated : the Packets Navigator is foreseen **only to consult** the packets in the Archive. The time consumed to open the Packets Navigator is only the one needed to find the first packet corresponding to the selected items list. For example, if the user has selected all the ADUs and GDUs from the Archive files, the Packets Navigator appears immediately.*

- Navigate through the packets with the navigation tool :

The user can inspect the packets only one by one. To compare some packets, open a second Raw Data Dump tool and start a second Packets Navigator. Up to four Raw Data Dump tools can be opened at the same time.

Press the **Next/Previous** button to display the next/previous packet in the Archive

OR

Type a time tag in the text field beneath the **Jump To** button.

Press the **Jump To** button to display the first packet for which the time tag is equal to or greater/less than the time tag specified.

OR

Type an index in the text field beneath the **Jump To** button (indices start at 1, a negative index counts from the end, i.e. –1 denotes the last packet, –2 the last but one packet, etc.).

Press the **Jump To** button to display the packet at the index specified.

Copyright per DIN 34

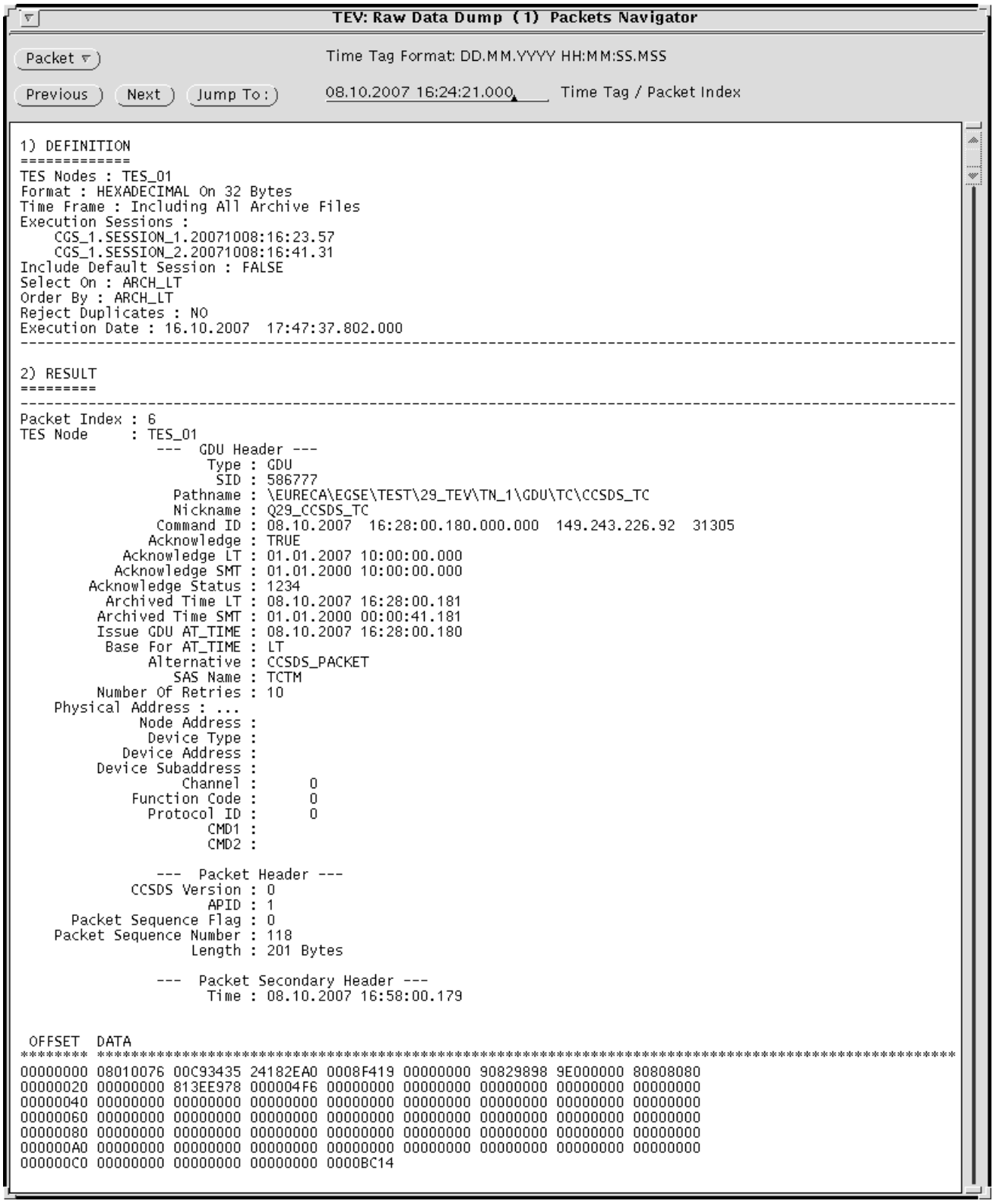

TEV: Raw Data Dump (1) Packets Navigator
Packet
Time Tag Format: DD.MM.YYYY HH:MM:SS.MSS
Previous
Next
Jump To:
08.10.2007 16:24:21.000
Time Tag / Packet Index
1) DEFINITION
TES Nodes : TES_01
Format : HEXADECIMAL On 32 Bytes
Time Frame : Including All Archive Files
Execution Sessions :
CGS_1.SESSION_1.20071008:16:23.57
CGS_1.SESSION_2.20071008:16:41.31
Include Default Session : FALSE
Select On : ARCH_LT
Order By : ARCH_LT
Reject Duplicates : NO
Execution Date : 16.10.2007 17:47:37.802.000
2) RESULT
Packet Index : 6
TES Node : TES_01
--- GDU Header ---
Type : GDU
SID : 586777
Pathname : \EURECA\EGSE\TEST\29_TEV\TN_1\GDU\TC\CCSDS_TC
Nickname : Q29_CCSDS_TC
Command ID : 08.10.2007 16:28:00.180.000.000 149.243.226.92 31305
Acknowledge : TRUE
Acknowledge LT : 01.01.2007 10:00:00.000
Acknowledge SMT : 01.01.2000 10:00:00.000
Acknowledge Status : 1234
Archived Time LT : 08.10.2007 16:28:00.181
Archived Time SMT : 01.01.2000 00:00:41.181
Issue GDU AT_TIME : 08.10.2007 16:28:00.180
Base For AT_TIME : LT
Alternative : CCSDS_PACKET
SAS Name : TCTM
Number Of Retries : 10
Physical Address : ...
Node Address :
Device Type :
Device Address :
Device Subaddress :
Channel : 0
Function Code : 0
Protocol ID : 0
CMD1 :
CMD2 :
--- Packet Header ---
CCSDS Version : 0
APID : 1
Packet Sequence Flag : 0
Packet Sequence Number : 118
Length : 201 Bytes
--- Packet Secondary Header ---
Time : 08.10.2007 16:58:00.179
OFFSET DATA
00000000 08010076 00C93435 24182EA0 0008F419 00000000 90829898 9E000000 80808080
00000020 00000000 813EE978 000004F6 00000000 00000000 00000000 00000000 00000000
00000040 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000060 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000080 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000A0 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
000000C0 00000000 00000000 00000000 0000BC14

Figure 8–21 : *Packet Navigator window showing a raw data packet*

The lower part of (NO TAG) is a scrollable text zone with the contents of the current packet. This subwindow is read–only.

Copyright per DIN 34

The format of the packets may be changed when the Packets Navigator is already open. Use the button **Packet:Edit Format ...** option from the Packets Navigator window or the **Edit Format** button from the main window. The *Output Format* pop–up window (Figure 8–22) appears :

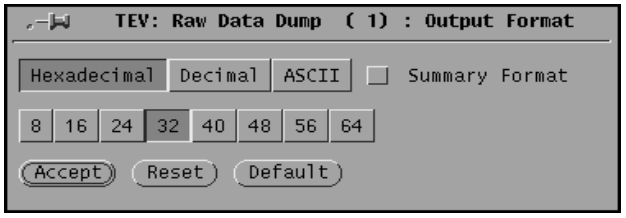


Figure 8–22 : *Output Format of a Raw Data Dump*

Select the Output Format for the Packets

- Press **Edit Format** button.
- Select the output base: Hexadecimal, Decimal or ASCII
- Select the number of bytes per line
- Press the **Accept** button.

*To go back to the default format (Hexadecimal, 32 bytes per line) click on the **Default** operation.*

The format specified is used for packets and complete dumps, when the data is a byte stream (i.e. for unstructured data, CCSDS packets).

The layout of the data dump may also be specified within the Packets Navigator :

Reformat a Raw Data Packet

- In the navigation tool press the **Packet** button and select **Packet : Edit Format...** from pop–up menu.
- Change the output format in the Output Format window (Figure 8–22)
- Press the **Accept** button in the Output Format window.

8.8.6.3 Consult the Packets in the Archive in Summary Format

The summary format is obtained through the **Edit Format** button of the Raw Data Dump main window. It allows the user to get an overall view of the packets dumped in the Archive.

Copyright per DIN 34

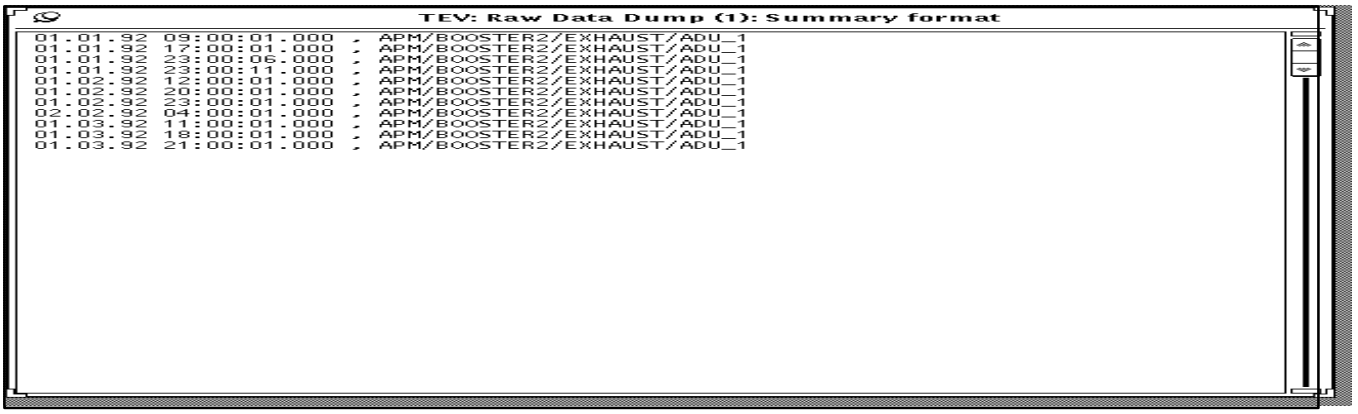

TEV: Raw Data Dump (1): Summary format
01.01.92 09:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.01.92 17:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.01.92 23:00:06.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.01.92 23:00:11.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.02.92 12:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.02.92 20:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.02.92 23:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
02.02.92 04:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.03.92 11:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.03.92 18:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1
01.03.92 21:00:01.000 ; APM/BOOSTER2/EXHAUST/ADU_1

Figure 8–23 : *A Summary Format*

Consult the Packets dumped in the Archive in Summary Format

- Create a Definition
- Click on **Edit Format** and choose in the window **Output Format** the option : **Summary Format**.
- Click on **Accept** in the window **Output Format**
- Select **Result : Start Packets Navigator** from the pop–up menu.

*A window pops–up and proposes an Interrupt button. TEV reads all the Archive Files, generates and displays the result. No result file is generated : use **Result : Save Dump in File** with the option **Summary Format** to generate a summary result file. Interrupt the dump when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. The result is generated until the interruption.*

The Raw Data Dump generation may be interrupted automatically because of a ”Disc Space problem”. In this case the result is generated until the interruption.

Copyright per DIN 34

8.8.6.4 Save the Selected Packets in a Result File

Save a Dump in a Result File

- Create a Definition.
- Choose a format using the **Format:** choice:
  - TXT (ASCII text)
  - ADT (binary)
- Select summary format:
  - **Edit Format:** check box **Summary Format**
- *No summary format possible in ADT format*
- Save the result: **Result:Save Dump In File...** The result file is generated in the directory **tev_working_directory/*RESULTS/RAW_DATA_DUMP*** (with the extension ".adt" when the ADT format has been selected).
- *A windows pops–up and proposes an Interrupt button.Interrupt the dump when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. The result is generated until the interruption.*
- *The Raw Data Dump generation may be interrupted automatically because of a "Disc Space problem". In this case the result is generated until the interruption. The result can be afterwards displayed normally by selecting **"Result:Load"** ... (Recover space before loading an ADT result file, because TEV creates an ASCII text file containing the formatted binary result).*

Copyright per DIN 34

8.8.7 Data Set Generation Tool

To access *Data Set generation* tool operations, the user must select the **Data Set generation...** item in the ***Tools*** menu :

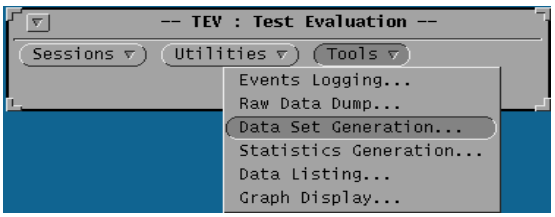


Figure 8–24 : *Start of the Data Set Generation Tool*

This tool allows the user to create data sets containing raw value and their associated calibrated engineering value. The values are extracted from either the archived files or from the engineering value logbooks.

The *Data Set Generation* main window is presented in Figure 8–25.

*Note that there is no **Result** menu as a Data Set is not directly displayed in this tool. Data Listing, Statistics and Graph tools are provided for this task.*

TEV allows to generate a data set, i.e. a set of values for selected measurements (maximum 50) within a specified time frame. Values may be sampled to take all from the time frame or only every nth one. The result is written to a Data Set file.

For more information about *Definition* operations refer to Chapter 8.8.1.
The upper part of the *Data Set Generation Window* (Figure 8–25) contains the Time Frame selection area, described into chapter 8.4.4.

The **TEV: Data Set** window :

Copyright per DIN 34

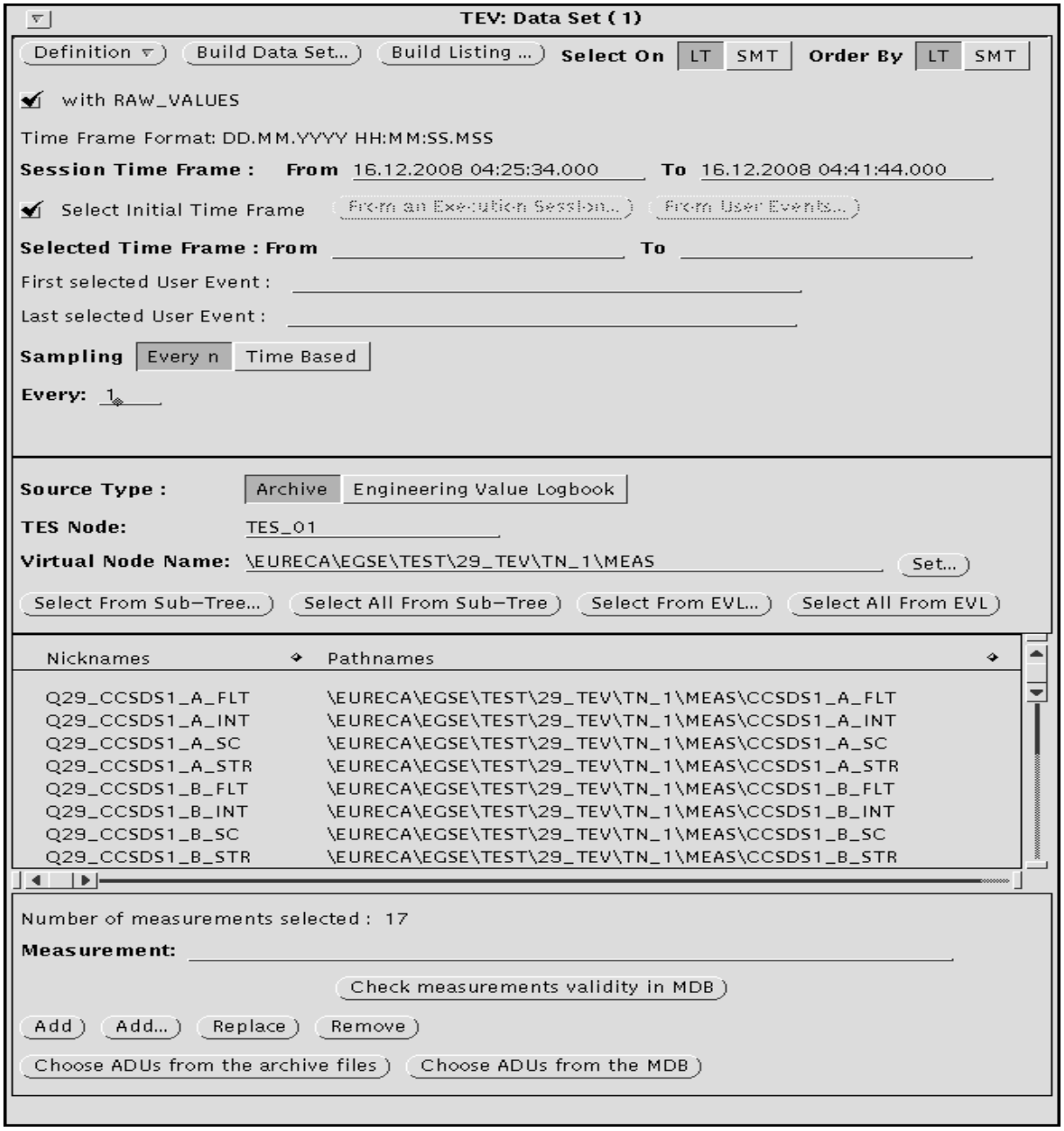


Figure 8–25 : *TEV: Data Set Generation Window*

Copyright per DIN 34

8.8.7.1 Select the Sampling Mode

Select the Every n Sampling Mode

- Press the **Sampling** menu button.
- Select **Sampling –> Every n** from the pop–up menu.
- Type the number for selecting every nth sample in the **Every: ____** text field.
- *If the user wishes to include all data the entry in the **Every**: field is 1.*

When ”n sampling” has been defined, TEV insert in the Data Set each nth value found for each parameter.

Select the Time based Sampling Mode

- Press the **Sampling** menu button.
- Select **Sampling –> Time Based** from the pop–up menu.
- Set up the **Time Interval** and **Allowed Error fields.**

Figure 8–26 shows the sample selection in the Time Based mode.

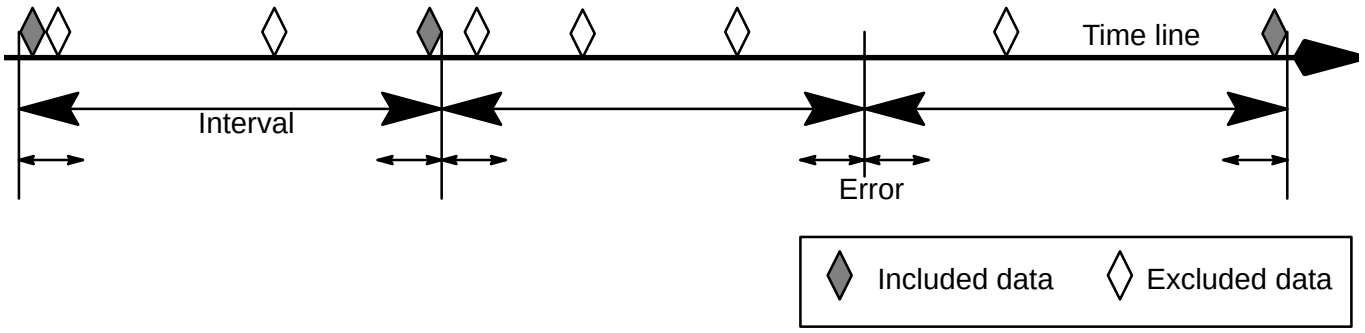


Figure 8–26 : *Time Based Sampling Semantics*

When a time based sampling has been defined in the global time frame [T0 .. T1] as : the time interval is DeltaT and the error is Eps, TEV extract the value for each parameter which is in the intervals :
– [T0 .. T0 + Eps] for the first value
– [T0 + mDeltaT – Eps .. min(T1, T0 + mDeltaT + Eps)] for the last value (m is the greatest number where T0 + mDeltaT <= T1)
– [T0 + iDeltaT – Eps .. T0 + iDeltaT + Eps] for the intermediate values (i = 1 .. m–1)

The Allowed Error interval assures that data which have a slight deviation at the time of their occurrence are included in the sample.A value is selected if it lies within the error limits.
If there are more than one value within the allowed interval, TEV extract the first value found in the interval.

8.8.7.2 Select Data Set Parameters

The parameters selected are the measurements from which the values have to be included into the Data Set.

Copyright per DIN 34

Define a Parameters Set

- Select the source type : **Archive** or **Engineering Value Logbook**

The Engineering Value Logbook contains those values which were already calibrated during the test, the archive files contain raw data.

Enter the name of the TES node that produced the data to be extracted in the **TES Node: ____** text field. By default, if no TES node is specified, data from all TES nodes involved in the selected execution sessions will be included.

A – Select Parameters from MDB

- Enter the node name in the **Virtual Node Name: ____** text field. This can be done either by typing in the name or via selecting the node using the MDB Chooser tool. To open the MDB Chooser tool, click on the **Set...** button to the right of the text field.

The Virtual Node Name defines the root pathname (Sub–Tree) in the MDB for the measurements to be selected.

- Click on the **Select From Sub–Tree...** button.
- Wait until the **TEV: Data Set**(i) :**Parameter Names** subwindow pops up.
- Select measurements from the list (Figure 8–27).
- Press the **Add** button.

The selected measurements appear in the tool main window, the subwindow **TEV: Data Set**(i) :**Parameter Names** *disappears. Duplicate entries are automatically ignored.*

OR

- Click on the **Select All From Sub–Tree** button.

All the measurements under the Virtual Node Name will be selected. Duplicate entries are automatically ignored.

B – Select Parameters from EVL

If the parameters to be included in the data set have been logged in the EVL, it may be more convenient to select them from the list of all parameters logged in the EVL during the currently selected execution session(s).

- Click on the **Select From EVL...** button.
- Wait until the **TEV: Data Set**(i) :**Parameter Names** subwindow pops up.
- Select measurements from the list (Figure 8–27).
- Press the **Add** button.

The selected measurements appear in the tool main window, the subwindow **TEV: Data Set**(i) :**Parameter Names** *disappears. Duplicate entries are automatically ignored.*

OR

- Click on the **Select All From EVL** button.

Copyright per DIN 34

- *All the measurements logged in the EVL during the currently selected execution session(s) will be selected. Duplicate entries are automatically ignored.*
- C – Select Parameters Manually
  - Enter a measurement name or a nickname into the **Measurement**:___ text field.
  - Click on the **Add** button.
- *The measurement is added into the list, if it is no already in the list.*
- Click on the **Check Measurements Validity In MDB** button : TEV will automatically check the existence and validity of the measurements selected in the currently displayed parameters set in the selected CCU. If some measurements are invalid the user shall remove them from the list.
- *In case a measurement is acquired by multiple ADUs, it is possible to select which ADUs shall be consider for the measurement :*
- Select a measurement in the list of selected measurements.
- Click on the **Choose ADUs From Archive Files...** button. The list of ADUs appearing in the archive files and which acquire the selected measurement appears. Click on the wanted ADUs in the list to select them.
- OR
- Click on the **Choose ADUs From MDB...** button. The list of all ADUs from the MDB which acquire the selected measurement appears. Click on the wanted ADUs in the list to select them.

- *The name of an already selected measurement can be modified. Select the measurement from the list : the name is copied into the Measurement field. Modify the measurement name and click on the **Replace** button (which is enabled only when a measurement is selected).*
- *Measurements may be removed from the list by selecting them and clicking on the **Remove** button.*

Note that selection into the list of selected measurements in the main window has no meaning for data set generation (***Build Data Set***), the pre–view listing generation (***Build Listing***) or for the parameter set checking (***Check Measurements validity in MDB***)

8.8.7.3 Configure Time Tags

Each sample in a data set is associated with two time tags: one for local time and one for mission time. However, CGS maintains up to 5 time tags for each sample: LT and MT from the archive (time data was archived by CGS), LT and MT from the ADU (time as set by SAS, usually time data was received by SAS), and possibly a packet time (MT from the CSSDS packet secondary header, onboard time packet was generated). On which of these time tags the data set generation should actually be based on can be defined via the attributes Data_System.Time_Stamp.Measurement_Time_Stamp_From_ADU and Data_System.Time_Stamp.ADU_Time_Stamp_From_CCSDS_Packet in the CGS configuration:

- Measurement_Time_Stamp_From_ADU = False
  ADU_Time_Stamp_From_CCSDS_Packet = False
  - the LT tags of all samples are taken from the archive

Copyright per DIN 34

- the MT tags of all samples are taken from the archive

- Measurement_Time_Stamp_From_ADU = False
  ADU_Time_Stamp_From_CCSDS_Packet = True
  - the LT tags of all samples are taken from the archive
  - the MT tags of all samples of measurements contained in CCSDS packets with packet time in secondary header are taken from the CCSDS packet secondary header
  - the MT tags of all other samples are taken from the archive
- Measurement_Time_Stamp_From_ADU = True
  ADU_Time_Stamp_From_CCSDS_Packet = False
  - the LT tags of all samples are taken from the ADUs
  - the MT tags of all samples are taken from the ADUs
- Measurement_Time_Stamp_From_ADU = True
  ADU_Time_Stamp_From_CCSDS_Packet = True
  - the LT tags of all samples are taken from the ADUs
  - the MT tags of all samples of measurements contained in CCSDS packets with packet time in secondary header are taken from the CCSDS packet secondary header
  - the MT tags of all other samples are taken from the ADUs

By default both attributes are set to False, thus LT and MT tags of all samples are taken from the archive.

To ensure duplicate CCSDS packets are ignored, generate the data set ordered by SMT with ADU_Time_Stamp_From_CCSDS_Packet set to True.

Copyright per DIN 34

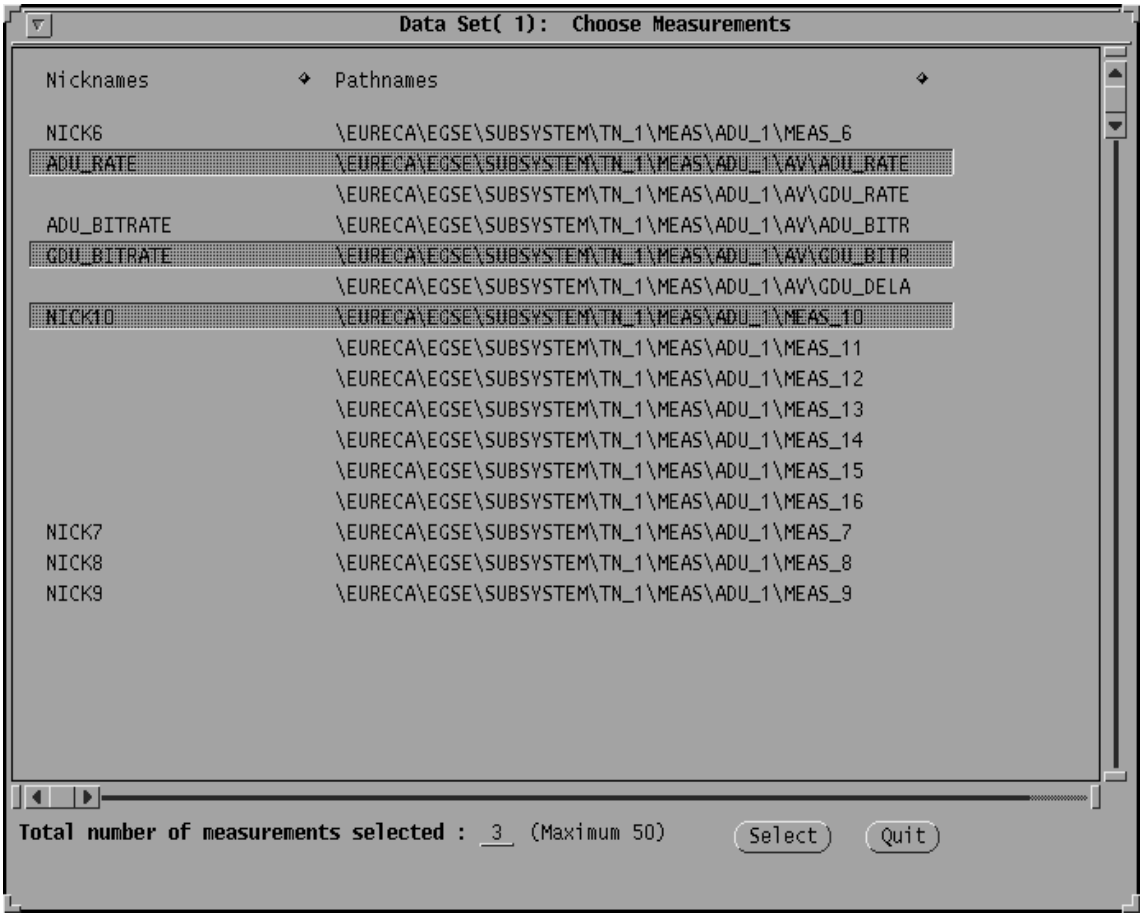


Figure 8–27 : *Selection of Parameters Below the Virtual Node Name Tree*

8.8.7.4 Generate a Data Set

Generate the Data Set

- Click on the **Build Data Set ...** button.
- Enter the new data set name.
- Click on the **Build** button
- A windows pops–up and proposes an Interrupt button
- *Interrupt the data set when the generation seems too long or when the disk space on $TEV_HOME is decreasing dramatically. In this case the user will get* ***no result****.*
- *The Data Set is generated in the directory* **tev_working_directory/*RESULTS/DATA_SET.***

When the generation is completed, the Data Set is available for statistic generation, data listing and graph display.

- *When some problems occured, TEV displays a text–editor containing a report about the generation. The file containing the report is in the directory* ***$CGS_HOME/local/tmp/tev/tmp_*machine_user.**

Copyright per DIN 34

The data sets merger tool allows two data sets that were built with the same time type criteria to be merged, so that data from different test sessions shall be mixed, ordered by time tags.

To access the Data Set Merger, the user must select the '**Data Set Merger...**' item in the *Utilities* menu. (See Chapter 8.5.2).

8.8.7.5 Generate a Pre–View Listing

The user may also display a pre–view listing in one shot, this means without Data Set generation. In this pre–view listing the data are not sorted, but displayed as extracted from the archive files (i.e in the same order as they were archived).

Generate a listing

- Click on the **Build Listing...** button.
- Enter the listing name.
- Click on the **Build** button
- *A listing is generated in the tev_working_directory/**RESULTS/DATA_LIST** directory and displayed.*

Copyright per DIN 34

8.8.8 Statistics Generation Tool (in CGS 6.3.2 replaced by Data Viewer Tool)

Note: To access the *Statistics Generation* tool in CGS 6.3.2, TEV must be started in backward compatibility mode.

To access *Statistics Generation* tool operations, the user must select **Statistics Generation...** item in the ***Tools*** menu (Figure 8–33) :

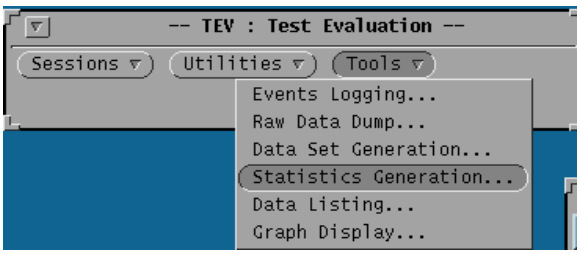


Figure 8–28 : *Start of the Statistics Generation Tool*

To generate a statistics result, it is not necessary to have selected some sessions. Statistics are generated from a Data Set. If Execution Sessions are selected, these ones are ignored.

- *Statistics Generation Tool (like the Data Listing tool and the Graph Display tool) works on data sets only.*

Statistics contain maximum and minimum values, mean, median, variance and the number of measurements.

For more information about *Definition* and *Result* operations refer to the Chapter 8.4.2 and 8.4.3

The *Statistics Generation* main window is presented in the next figure :

Copyright per DIN 34

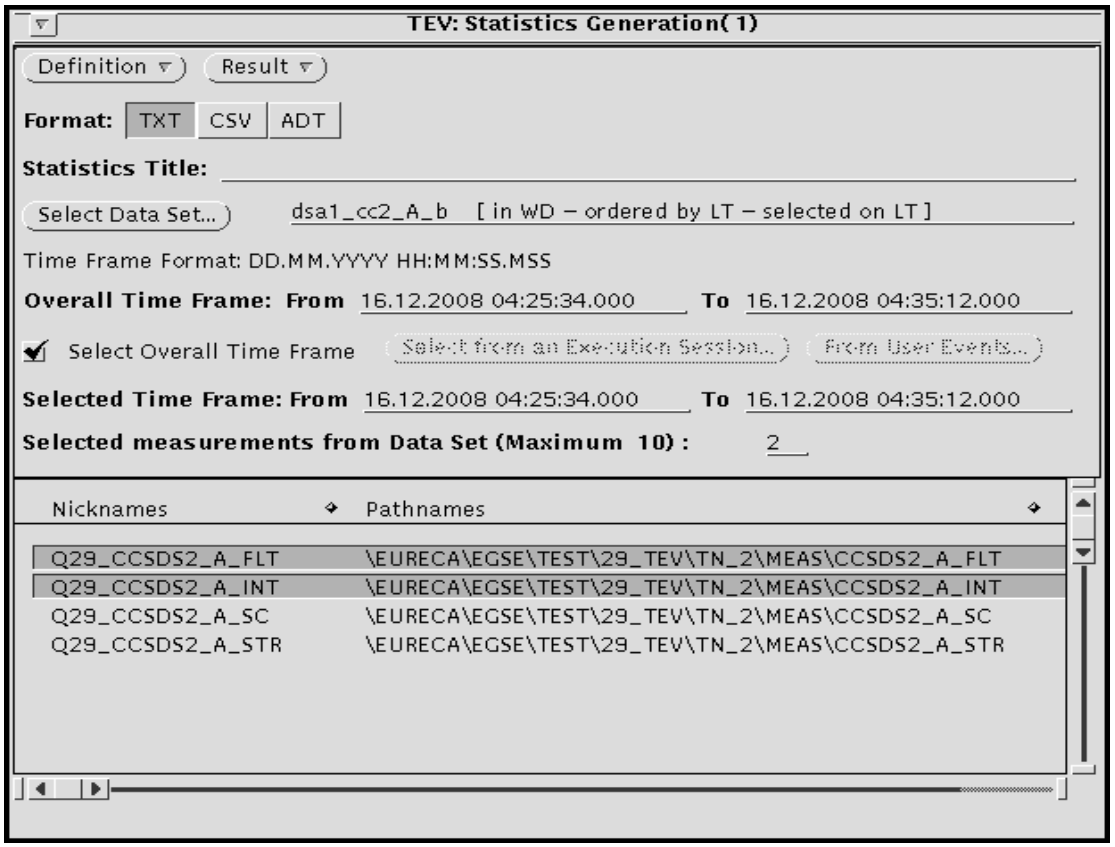


Figure 8–29 : *The Statistics Generation Tool Window*

Copyright per DIN 34

Build a Statistics Generation Definition

- Type the name of the statistic in the **Statistics Title: ____** text field. This name will be included in the result file.
- Click on the **Select Data Set...** button.
- Select the desired data set in the window ***TEV: Statistics Generation: Data Set Results.***
- Press the **Select** button.
- The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the Data Set selected.
- *By default the **Select All Parameters** check box is enabled : once the data set has been selected the first 5 measurements from that data set will be displayed in the parameters list. If the data set contains more than 5 parameters (A data set may have up to 50 measurements) a message to indicate that the list has been truncated will appear in the footer of the graph display window.*
- Disable the **Select All Parameters** check box.
- Press the **Select Parameters...** button.
- *The window **Statistics Generation (i) : Data Set Parameters** appears, containing the list of all the measurements included into the selected Data Set.*
- Select the desired measurements from the list.(Up to 5).
- Press **Select** button. The reduced list is displayed in the **Statistics Generation** window.
- Reduce the time frame to limit the data included in the statistics : disable the **Select Overall Time Frame** box and select a time frame either form an execution session, either from User Events.

Once the wanted Statistics Generation Definition is defined, click on **Result–> Exec&Display** to produce the result.

Copyright per DIN 34

8.8.9 Data Listing Tool

To access *Data Listing* tool operations, the user must select the **Data Listing...** item in the *Tools* menu.

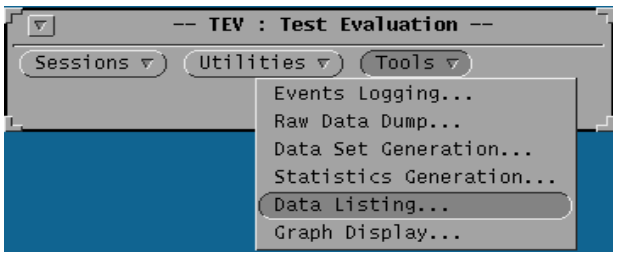


Figure 8–30 : *Start of the Data Listing Tool*

The results generated here do not depend on the selected execution sessions, but on a selected data set. Before the user can generate a data listing the actions described in Chapter 8.8.7 must be performed, e.g. a data set has to be created.

The Data Listing tool (like the Statistics Generation tool and the Graph Display tool) works on data sets only.

For more information about *Definition* and *Result* operations refer to the Chapter 8.4.2 and 8.4.3.

The Data Listing tool is very similar in use to the Statistics Generation tool, the structure of the generated result only differs. Once a data set has been selected, The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the data set from which data is extracted.
The procedure to select measurements from the data set is the same.

The following formats can be specified via the ***Format:*** choice:

- ***TXT*** : TEV generates a listing result in plain text format. The file is generated in **tev_working_directory*/RESULTS/DATA_LIST*** without extension.
- ***ADT*** : TEV generates a listing result in a TEV internal binary format. This file may be read and written externally within an Ada program using the TEV_API library. The file is generated in **tev_working_directoy/*RESULTS/DATA_LIST*** with extension .adt.
  A file in ADT format can be loaded and displayed as plain text using the menu command “***Results:Load”***.
- ***CSV*** : TEV generates a listing result in CSV format. The file is generated in **tev_working_directory*/RESULTS/DATA_LIST*** with extension .csv.
- ***LACIS*** : A project specific format. The file is generated in **tev_working_directory*/EXTERNAL/DATA_LIST***.

Copyright per DIN 34

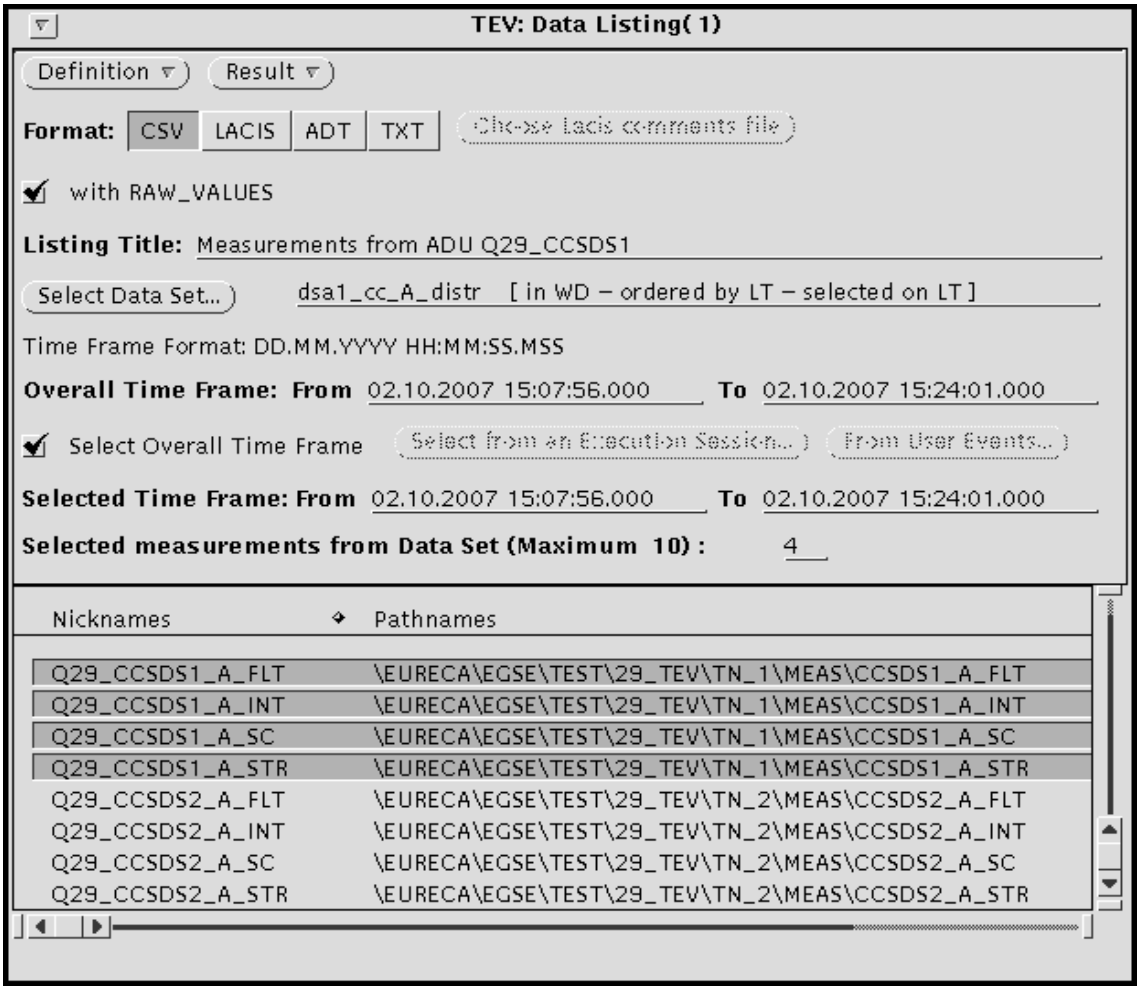


Figure 8–31 : *TEV Data Listing window*

If the option ***With Raw Values*** is selected, TEV generates a listing result with engineering values and raw values for each measurement selected.

Byte stream measurement values are truncated in a result listing in TXT format, except if the byte stream measurement is the ONLY ONE measurement selected. In this case, the measurement value is displayed completely.

In CGS 6.3.2 the CSV format for data listings has become a TEV internal format, the file is generated in **tev_working_directory/*RESULTS/DATA_LIST*** instead of, as before, in **tev_working_directory/*EXTERNAL/DATA_LIST***. The **Result Files List** dialog opened when invoking the **Result** menu's *Load...* or *Save...* operation, now includes these CSV files indicating them with the extension *.csv*. The *Load...* operation on a data listing result file in CSV format now opens the new Data Viewer tool. The Data Viewer is also opened when the **Result** menu's *Exec & Display...* operation is executed if the format option has been set to *CSV*.

8.8.10 The *Result* Menu

This menu offers the operations on Evaluation Result Files :

The *Load...* operation reads a result file and displays it into a specific window. The title of this window contains the tool reference, its instance number and the result file name. The result file can be loaded either from

Copyright per DIN 34

the directory tev_working_directory/*RESULTS/DATA_LIST* (*source:Working Directory*), or from the current open Evaluation Session (*source:Test Result Data Base*).

The *Save...* operation saves the current result into the directory tev_working_directory*/RESULTS/DATA_LIST*. (To save a result into the current open Evaluation Session, use the *Store* operation from the File Manager tool. See Chapter 8.5.1).

The *Exec&Display...* operation creates a result for the current definition. The definition is first checked for error. In case of error no execution takes place. During execution, further operations in the window are disabled and this is shown by the shaded title–bar and stopwatch pointer. However, operations in other windows are permitted.
If the window has been unpinned and the definition has not been changed since the last execution, invoking the *Exec & Display...* operation will not generate the result again but simply display the previously generated result.
When the execution is complete, the result is displayed in a separate result window. The title of this window contains the tool reference and its instance number. The user can browse at the result in the usual OpenWindows way.

Exec & Display a Definition

- Select **Definition –> Load...** from the pop–up menu.
- Click on the definition name in the list.
- Press the **Load** Button.
- Select **Result –> Exec & Display** from pop–up menu.

Copyright per DIN 34

Build a Statistics Generation Definition

- Type the name of the statistic in the **Statistics Title: ____** text field. This name will be included in the result file.
- Click on the **Select Data Set...** button.
- Select the desired data set in the window ***TEV: Statistics Generation: Data Set Results.***
- Press the **Select** button.
- The *Initial Time Frame* is replaced by the *Overall Time Frame* which is the overall time frame of the Data Set selected.
- *By default the **Select All Parameters** check box is enabled : once the data set has been selected the first 5 measurements from that data set will be displayed in the parameters list. If the data set contains more than 5 parameters (A data set may have up to 50 measurements) a message to indicate that the list has been truncated will appear in the footer of the graph display window.*
- Disable the **Select All Parameters** check box.
- Press the **Select Parameters...** button.
- *The window **Statistics Generation (i) : Data Set Parameters** appears, containing the list of all the measurements included into the selected Data Set.*
- Select the desired measurements from the list.(Up to 5).
- Press **Select** button. The reduced list is displayed in the **Statistics Generation** window.
- Reduce the time frame to limit the data included in the statistics : disable the **Select Overall Time Frame** box and select a time frame either form an execution session, either from User Events.

Copyright per DIN 34

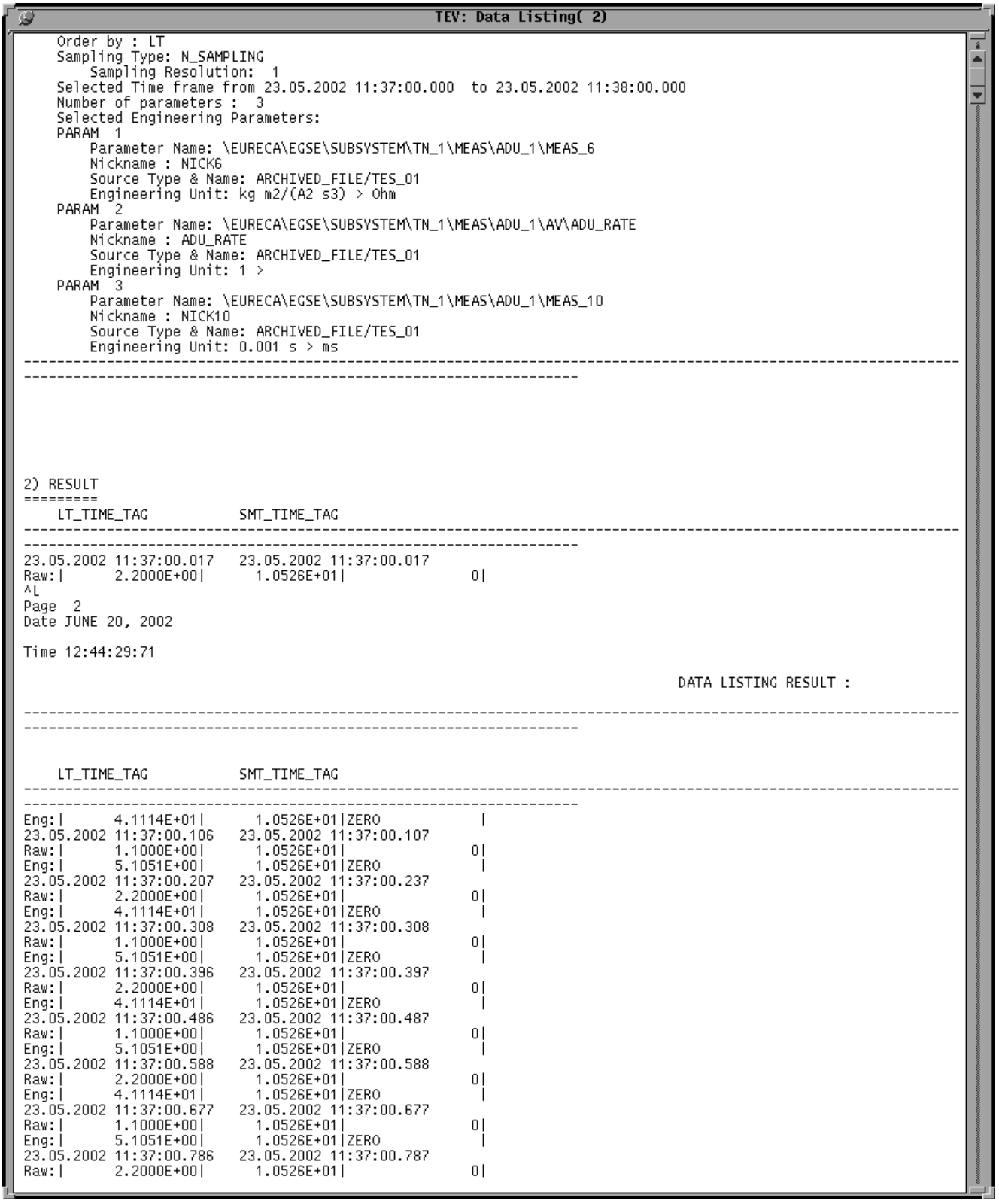


Figure 8–32 : *The Data Listing Generation Tool Window*

Copyright per DIN 34

8.8.11 Graphic Tool (in CGS 6.3.2 replaced by Data Viewer Tool)

Note: To access the *Graphic* tool in CGS 6.3.2, TEV must be started in backward compatibility mode.

To access *Graph* tool operations, the user must select the **Graph Display...** item in the ***Tools*** menu.

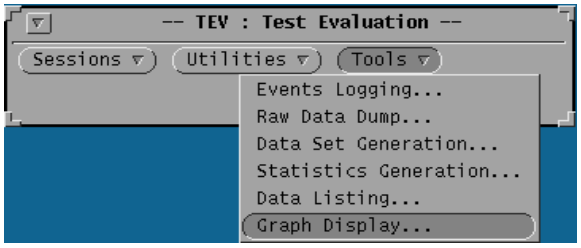


Figure 8–33 : *Start the Graph Tool*

The results generated here do not depend on the selected Execution Sessions, but on a selected Data set. Before the user can perform a Graph the actions described in Chapter 8.8.7 must be performed, e.g. a Data Set has to be created.

- *The Graph Tool (like the Statistics Generation tool and the Listing Generation tool) works on data sets only.*

For more information about *Definition* and *Result* operations refer to the Chapter 8.4.2 and 8.4.3.

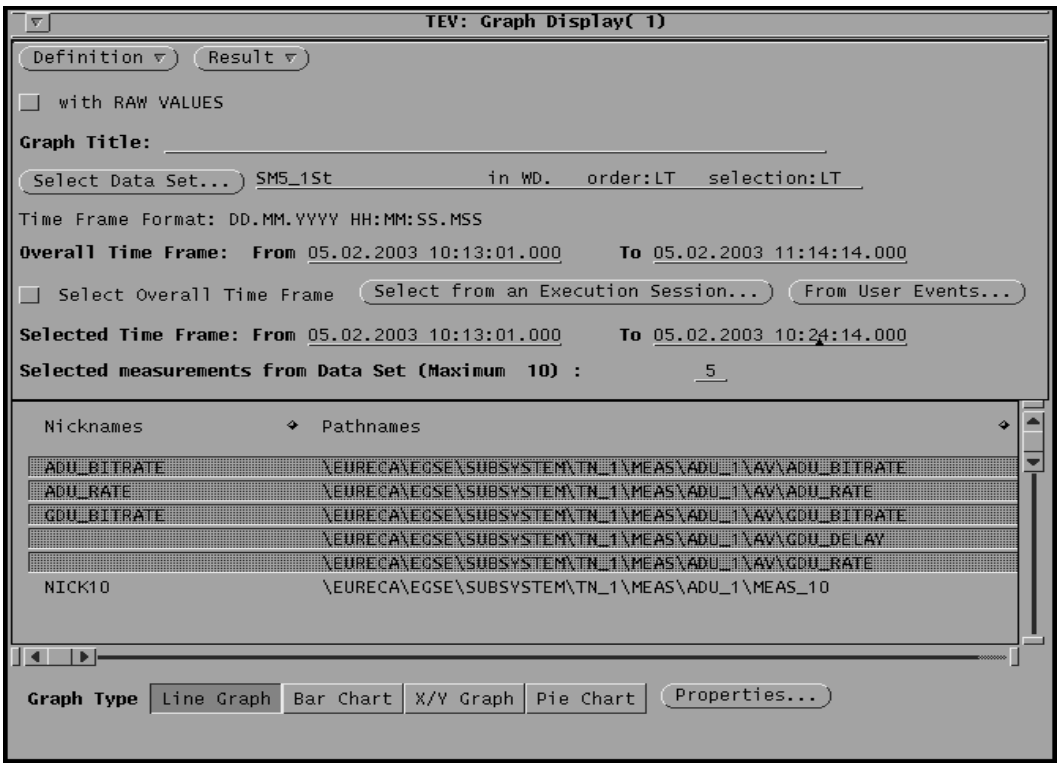


Figure 8–34 : *The Graph Tool Window*

Copyright per DIN 34

At the bottom of the window the **Graph Type** choice boxes allow the user to select the type of graph for the current definition.

Build a Graph Display Definition

- Type the name of the graph in the **Graph Title: ____** text field. This name will be included in the result file.
- Click on the **Select Data Set...** button.
- Select the desired data set in the window ***TEV: Graph Display: Data Set Results.***
- Press the **Select** button.
  The Initial Time Frame is replaced by the Overall Time Frame which is the overall time frame of the selected Data Set.
- *By default the **Select All Parameters** check box is enabled : once the data set has been selected the first 5 parameters from that data set will be displayed in the parameters list. If the data set contains more than 5 parameters a message to indicate that the list has been truncated will appear.*
- Disable the **Select All Parameters** check box.
- Press the **Select Parameters...** button.
- *The window **Graph Display (i) : Data Set Parameters** appears, containing the list of all the measurements included into the selected Data Set.*
- Select measurements from the list.
- *The graph tool prevents more than 5 parameters being selected from this list.*
  For X/Y Graph type TEV expects only 2 measurements.
  *If the option **With Raw Values** is selected TEV expects only one measurement.*
- Press **Select** button. The list of selected measurements is displayed in the Graph tool main window.
- *Selection in this list has no meaning.*
- Reduce the time frame to limit the data included in the graph : more than 200 points in the graph make this one hardly readable. For this disable the **Select Overall Time Frame** box and select a time frame :
  . either from an execution session
  . either from User Events
  . or enter directly a time frame in the fields
- Set the check box **With Raw Values.**
- *The option **With Raw Values** concerns only the Line Graph type. (Only one selected measurement is allowed). The raw values are implicitely used for measurements of type discrete, in this case the option **with Raw values** has no effect.*
- Select the **Graph Type** to be displayed.
- Click on the **Properties** button to define the graphical properties for the type of graph chosen.

There are four types of graph which the user may select :

Copyright per DIN 34

- **Line Graph**. The Line Graph is a graph of up to 5 measurements against either absolute time or relative time. Raw Values can be plotted into a line graph for one measurement. In this case 2 curves are represented : the one for the engineering values and the one for the raw values.
  For this the user shall select the box ***With RAW VALUES*** and one and only one measurement which is not of type DISCRETE or BYTE_STREAM. DISCRETE measurements are automatically displayed using the raw values.
- **Bar Chart**. The Bar Chart is calculated for the first value found in the time frame specified for each parameter selected.
- **XY Graph**. The XY Graph is a graph plotted of parameter x against parameter y. Use the ***Properties*** window to specify which one from the two selected parameters shall be plotted on the x axis.
- **Pie Chart**. The pie chart is calculated for the first value found in the time frame specified for each parameter selected.

For a **pie chart** at least two parameters must be defined. The first value to be found within the time period defined will be charted for each parameter. The values must be of the same sign and type or an error will be reported.

Edit the Line Graph Properties

- Press the **Properties...** button.
- Type the X Axis lettering in the **X Axis name:___** text field (max. 20 characters) (see Figure 8–35).
- Enable the **Time Frame** selection check box to **RELATIVE** or **ABSOLUTE.**
- *If RELATIVE, then the user can enter for each parameter a relative time frame.*
- Type the Y Axis lettering in the **Y Axis name:___** text field (max. 20 characters).
- Enable the **Scaling** selection check box to AUTOMATIC or MANUAL.
- If Scaling mode is MANUAL :
- Type the lower limit value in the **From:___** text field.
- Type the upper limit value in the **To:___** text field.
- *Values outside the range selected will not be plotted.*

The window *Line Graph properties* presents five identical paragraphs, each paragraph (i) allow to define the properties for the parameter (i) :

Copyright per DIN 34

- For each parameter:
- Press the **Colour** button with the right mouse button and select a colour from the pop–up menu.(see Figure 8–36)
- Press the **Line Style** button with the right mouse button and select a style from the pop–up menu.
- Press the **Thickness** button with the right mouse button and select the line thickness from the pop–up menu.
- If the RELATIVE Time Frame selection is enabled :
- Type the selection start time in the **Relative Time Frame From:___** text field.
- Type the selection stop time in the **Relative Time Frame To:___** text field.
- *The to and from fields control the data selection. TEV displays the first value selected for each curve at reference start time frame and displays subsequent values relative to this up to the end time.*

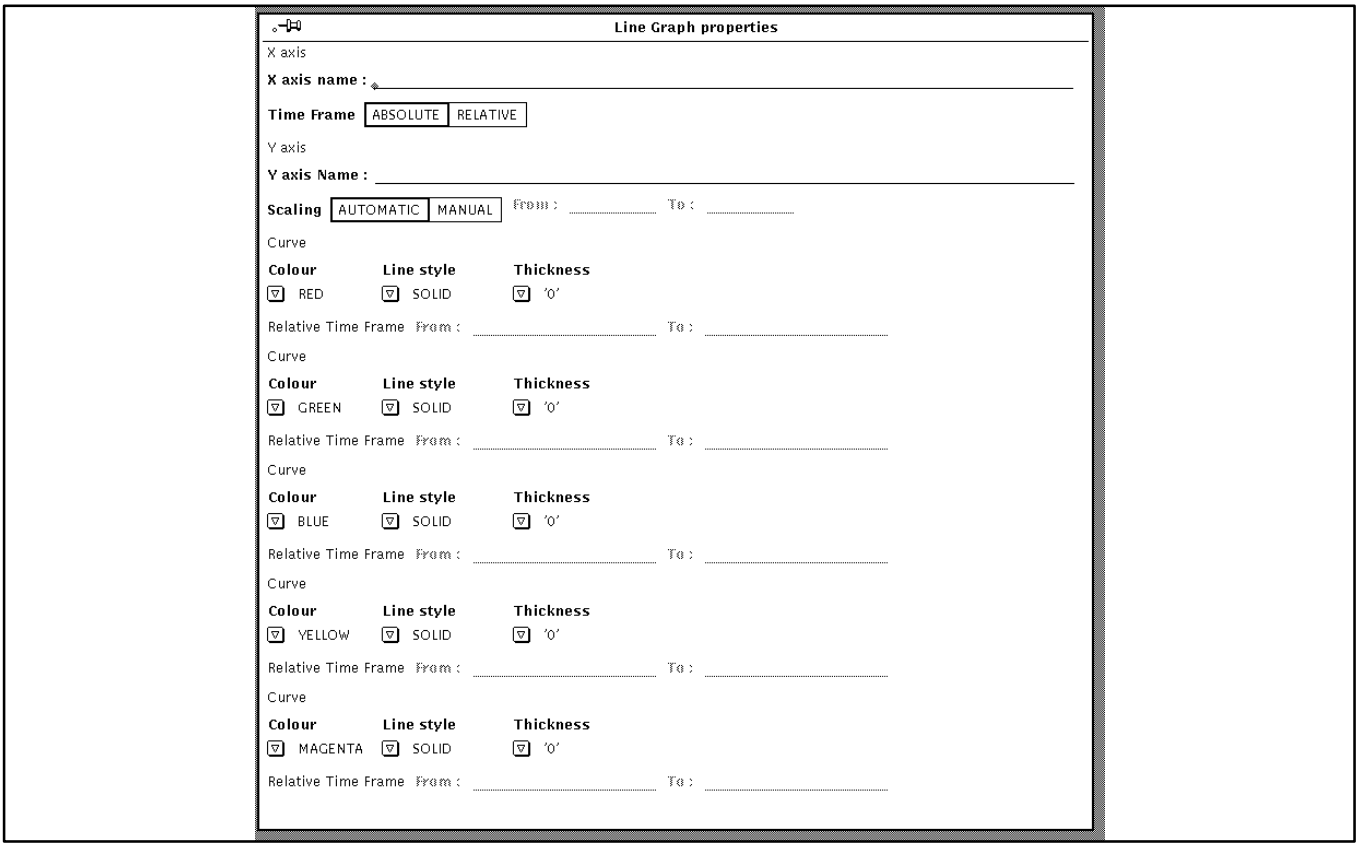


Figure 8–35 : *Line Graph Properties Window*

Copyright per DIN 34

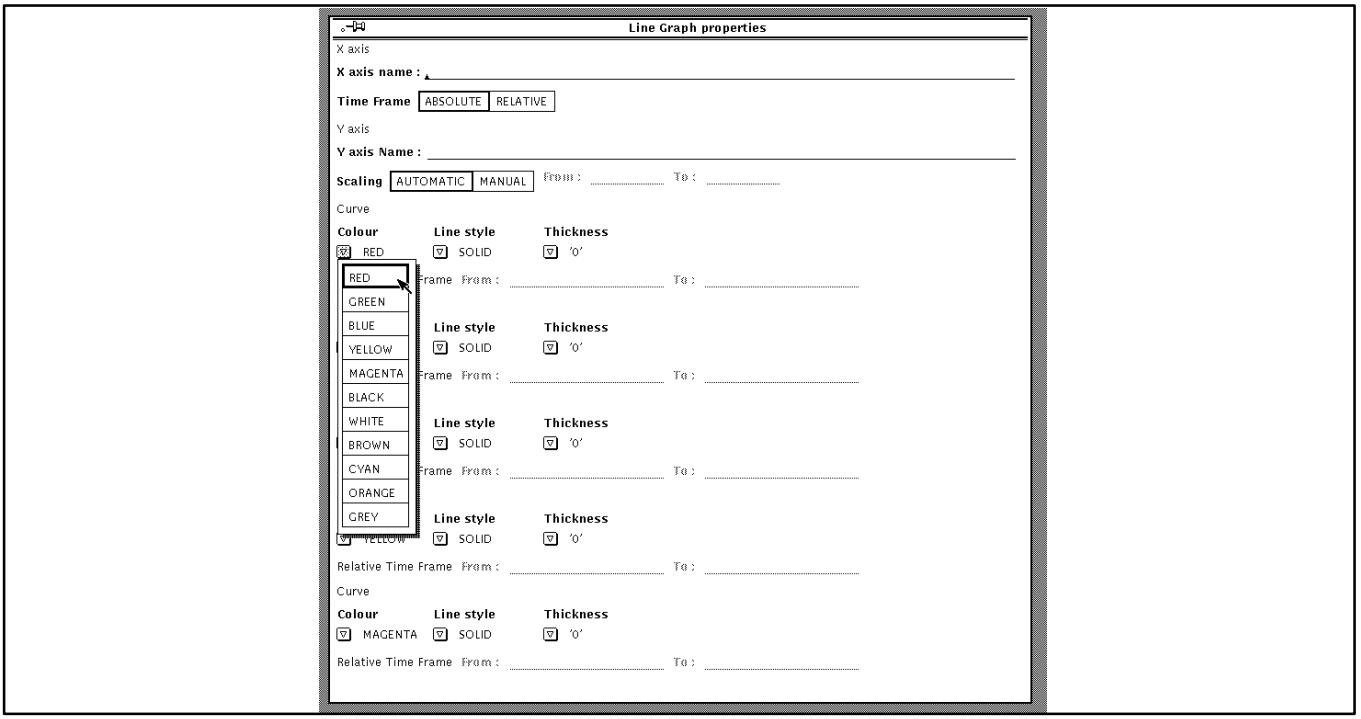


Figure 8–36 : *Line Graph Properties Window with Colour Selection*

The **Bar Chart** Properties window comprises only a colour menu which allows to select up to 11 different colours for each parameter.

Edit the XY Graph Properties (see Figure 8–37)

- Click on the Graph Type selection button **XY Graph**.
- Press the **Properties...** button.
- Choose which measurement shall be considered as X
- Type the X Axis lettering in the **X Axis name:___** text field.(max. 20 characters)
- Type the Y Axis lettering in the **Y Axis name:___** text field. (max. 20 characters)
- Enable the **Manual Scaling** check box for the X axis.
- Type the lower range in the **From:___** text field.
- Type the upper range value in the **To:___** text field.

Copyright per DIN 34

- *Values inside the range selected will be indicated plotted.*
- Enable the **Manual Scaling** check box for the Y axis.
- Type the lower range in the **From:___** text field.
- Type the upper range value in the **To:___** text field.
- *Values inside the range selected will be plotted.*
- Click on the **Colour** button and select a colour from the pop–up menu.
- Click on the **Line Style** button and select a style from the pop–up menu.
- Click on the **Thickness** button and select the line thickness from the pop–up menu.

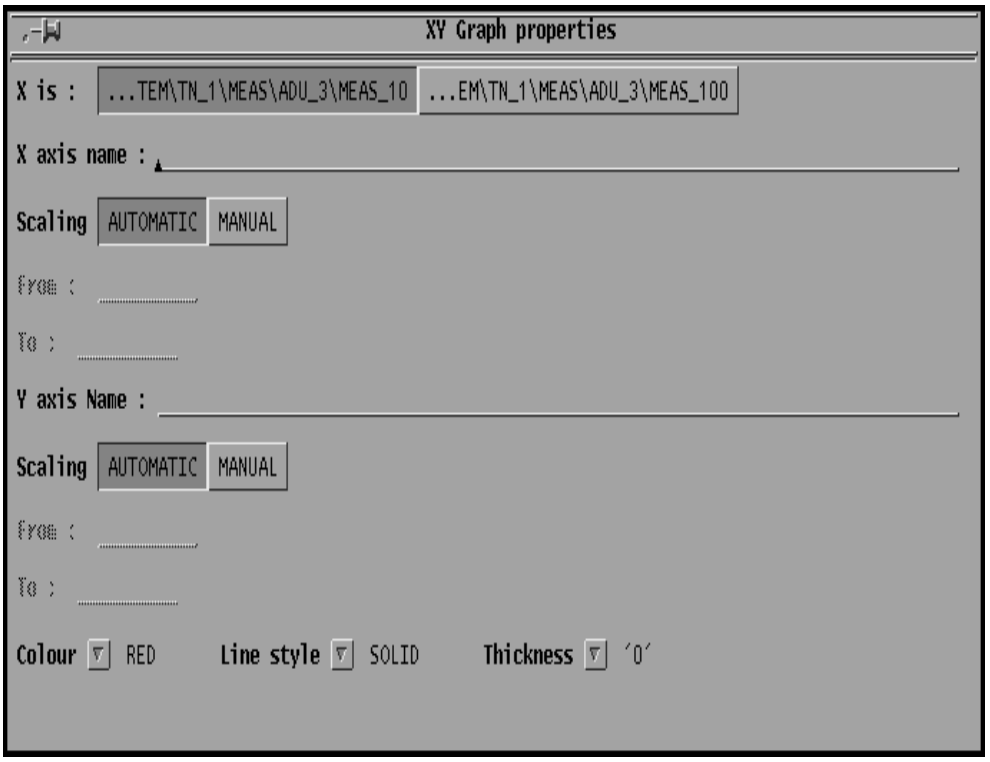


Figure 8–37 : *X/Y Graph Properties*

The Pie Chart properties window comprises a colour menu (see Figure 8–38) to select a colour for each parameter.

Copyright per DIN 34

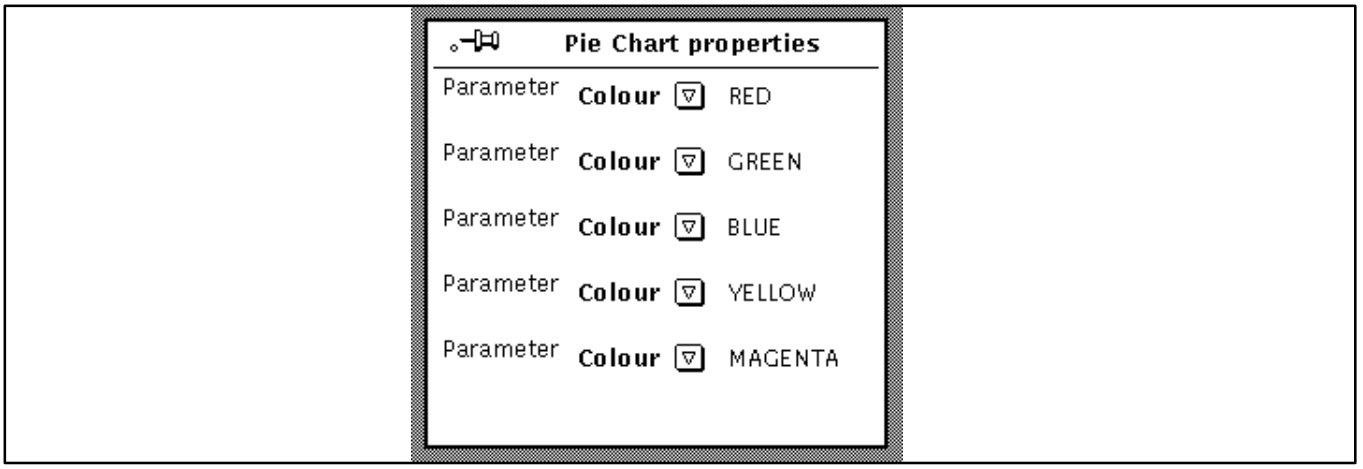


Figure 8–38 : *Pie Chart Properties*

Create a Graphic

- Press the **Result...** button in the main window.
- Select the **Exec & Display** option from the pop–up menu.

A graph result is a postscript file which may be displayed through TEV. The postscript file is generated in the directory **$TEV_HOME/data/tmp/tmp_*machine_username.*** When the graph is saved (**Result –> Save ...**), the poscript file is saved in **$USER_HOME/wd/tev/RESULTS/GRAPHS.** The generation of the TEV graph is pre–conditioned by the existence of the file **$TEV_HOME/data/tev_graph_prolog.ps.**

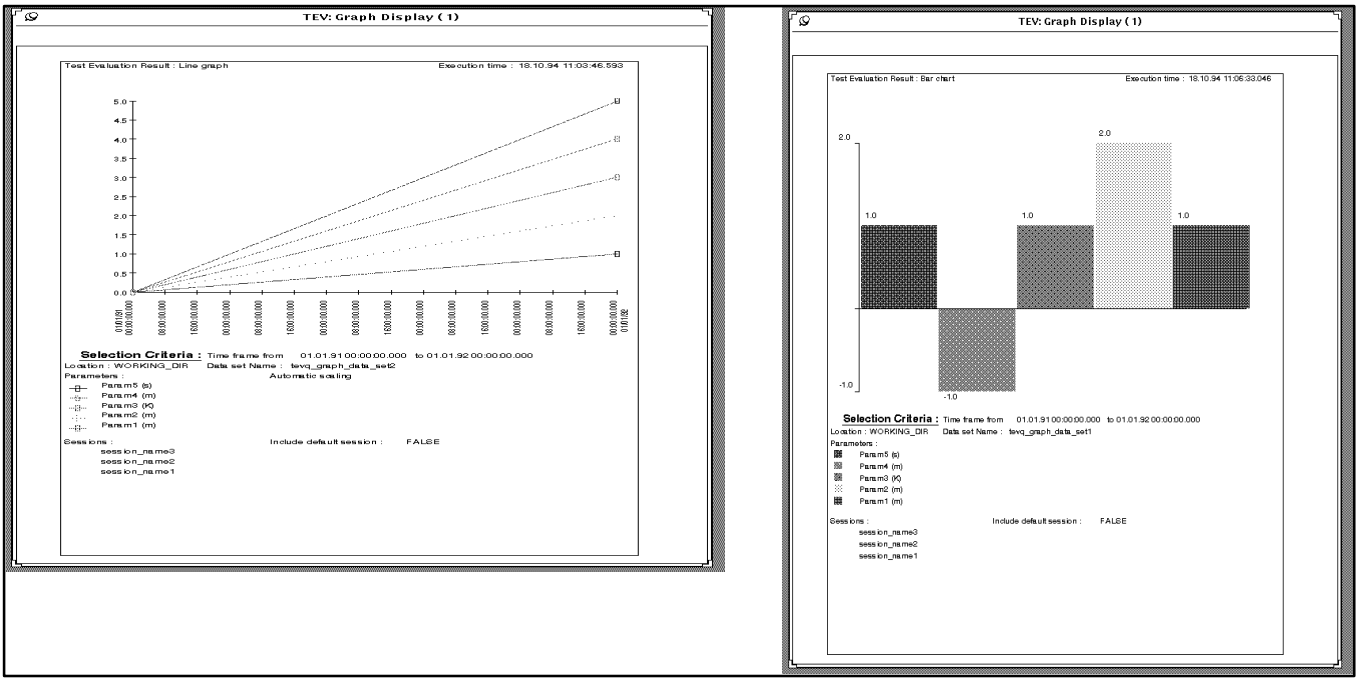

TEV: Graph Display (1)
Test Evaluation Result : Line graph
Execution time : 18.10.94 11:03:46.593
Selection Criteria :
TEV: Graph Display (1)
Test Evaluation Result : Bar chart
Execution time : 18.10.94 11:06:33.046
Selection Criteria :

Figure 8–39 : *Examples of Line Graph and Bar Chart*

Copyright per DIN 34

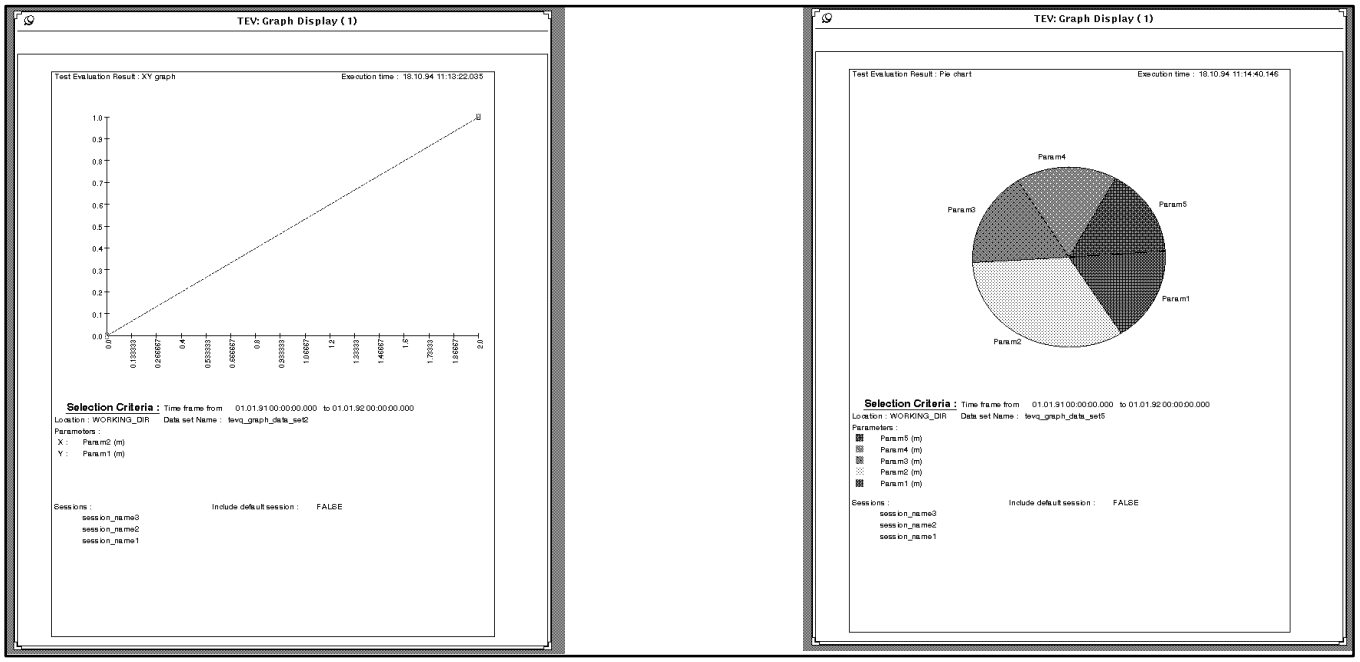


Figure 8–40 : *Example of XY Graph and Pie Chart*

Copyright per DIN 34

8.9 Utilities

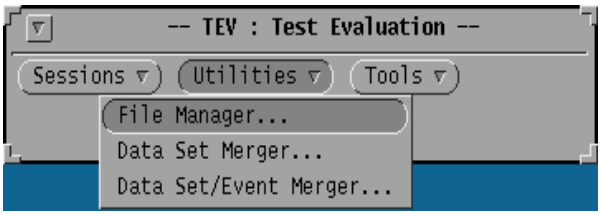


Figure 8–41 : *Utilties Menu*

The Utilities Tool offers several miscellaneous tools :

- The File Manager to manage the files which are the results of the User evaluations : Events Listing, Raw Data Dump, Data Sets, Graphs, Statistics, Data Sets Listings. These files can be deleted, renamed or copied in any directory. They can be stored in an Evaluation Session. Data Sets can be converted in Excel format.
- The Data Set Merger allows to merge two Data Sets, so that for example the data from two different sessions shall be mixed.
- The Data Set / Events Listing Merger allows to mix events and engineering values together, ordered by time tags.

8.9.1 File Handling

The File Manager allows the user to perform operations on result files in the working directory. The working directory is located in the users UNIX file system : tev_working_directory/RESULTS. All the results of evaluation are by default stored in the working directory.

When the user explicitely wants to store a result file in an evaluation session, the result file is then stored in the TRDB. The user has no possibility to set–up a mode, which would implicitely store all the result files in the current selected evaluation session. The user shall explicitely use the **Store** button from the **File Handling** Utilities and an evaluation session shall be open.

To access the File Manager tool the user must select the **File Manager** entry in the ***Utilities*** menu.

The **TEV : File Manager** (see next figure) window is made of :

- a **Result Type** selector, only one tool type can be selected at a time
- a list of file names existing in the tev_working_directory*/RESULTS* directory, that match the selected result type
- a counter field *'File(s) Selected'* giving the number of selected files in the list
- a group of four buttons: **Rename, Store, Delete** and **Copy**

Copyright per DIN 34

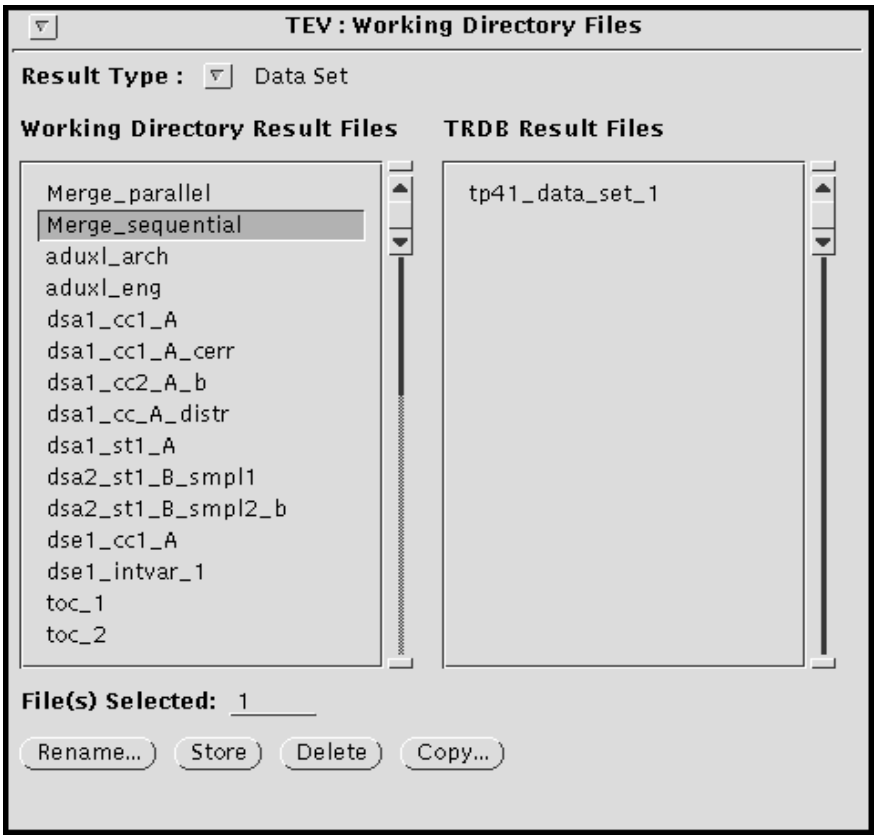


Figure 8–42 : *The Working Directory Files Manager*

- *In case the contents of the working directory or the TRDB has changed (due to work in another tool in another window) and is not up–to–date, clicking in the result type choice boxes will refresh the displayed list.*

Rename and **Store** options work on single files only. **Delete** and **Copy** functions work on one or more files.

Store a File in the Test Result Database (i.e in a Test Evaluation Session)

- Perform the procedure: **Select an Evaluation session** (see Chapter 8.7.1).
- Press one of the **Result Type** buttons to determine the type of result files to be listed.
- Scroll the file list by using the scrollbars, if necessary.
- Move the mouse into the list and select the desired file.
- Press the **Store** button.

Copyright per DIN 34

Delete one or more Files from the Working Directory

- Press one of the **Result Type** buttons to determine the type of result files to be listed.
- Scroll the file list by using the scrollbars, if necessary.
- Move the mouse into the list and select the desired file(s).
- Press the **Delete** button.
- Confirm the deletion in the confirmation notice window.

The ***Rename*** operation changes the name of a file. The new name is prompted in a command window.

A default name is proposed in this case (the current name of the file). The user can edit the name in the text field; when he is satisfied with the new name, clicking on the **Rename** button will perform this operation. If the new file name corresponds to a file that already exists, a confirmation notice appears to confirm **Overwrite** or **Cancel.**

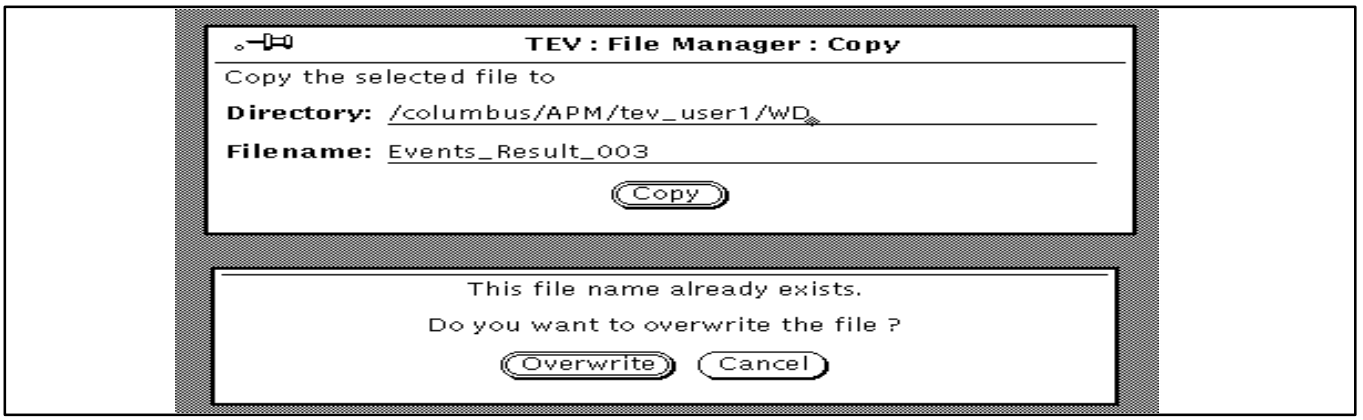


Figure 8–43 : *The Command Window to Copy a File plus Confirmation Window*

Copy a Single File

- Select a file name from the list.
- Click on the **Copy...** button, a pop–up window appears with two editable text fields and a button labeled **Copy** (see Figure 8–43).
- Type the destination directory in the **Directory** field.
- Enter the new filename in the **Filename** field.
- Click on the **Copy** button.
- If the given pathname corresponds to a file that already exists, a confirmation notice appears.
- Press the **Overwrite** button.

The feature to copy a single file allows to determine a new name for the file.

- *If you copy several files they are copied under their original names, i.e. source and destination names are the same.*

Copyright per DIN 34

Copy Several Files At Once

- Select the result names from the list.
- Click on the **Copy...** button.
- The command window Figure 8–43 appears.
  Note that only the **Directory** field is editable.
- Enter the destination directory.Click on the **Copy** button.

The copy operates from the first (topmost) selected file to the last. If an error is encountered, the files which have been copied successfully are deselected and the ones not yet copied remain selected. The user may choose to abort the operation or retry with the files which are remaining selected.

8.9.2 Data Set Merger

The data sets merger tool allows two data sets that were built with the same time type criteria to be merged

To access the Data Set Merger, the user must select the '**Data Set Merger...**' item in the *Utilities* menu.

The **TEV : Merge Data Sets** window is shown. From this window, the user is able to merge two data sets. The user should click on the ***First*** data set choice box to select the first data set and parameters. He should click on the ***Second*** data set choice box to select the second data set. The window toggles between the two data sets.

The following cases will cause error messages :

– If the engineering units for a selected parameter which is in both data sets are not the same

– If a selected parameter in both data sets has different values for the same time tag

– If a selected parameter in both data sets has different sources, i.e. a mix of engineering values log-books and archived files.

Once all the parameters and their time frame are selected for each data set, by pressing the **merge...** button, TEV will build a new data set file containing data extracted from both selected data sets. The user first has to give a name for this new data set file inside a pop–up window.

TEV will not allow the user to merge a data set onto itself. An appropriate error message will be given and the merge will not take place. The merge will also be refused if the resulting data set would contain more than 10 sessions.

8.9.3 Data Set / Events Merger Tool

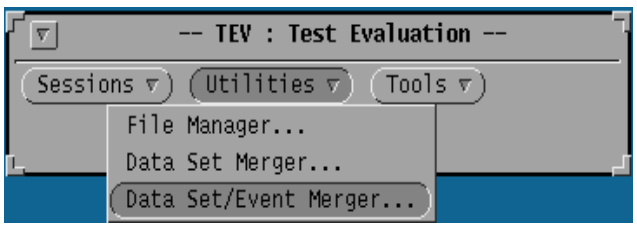


Figure 8–44 : *Start the Data Set/Events Merger*

Copyright per DIN 34

This tool allows the user to mix measurements values which are stored in a Data Set with the events stored in an Events Listing and to order them by time tags. This tool is only thougth to handle Events Listings concerning one session only.

To access this tool the user must select the ***Data Set/Events Merger*** entry in the ***Utilities*** menu.
The window ”TEV:Merge Data Set/Events” appears :

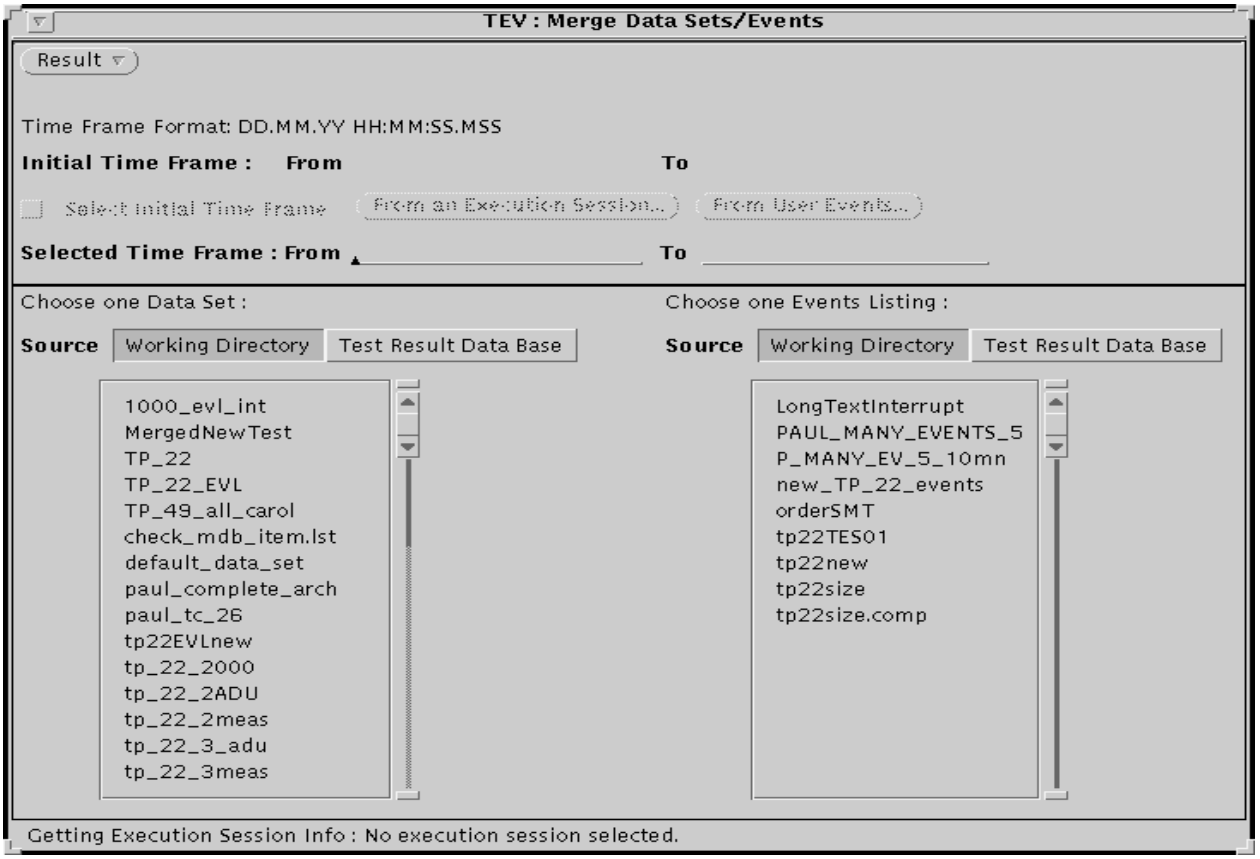


Figure 8–45 : *When No Session Selected .....*

If no session was previously selected, the option ***”Select Initial Time Frame”*** is not available. The ***”Selected Time Frame***” value depends on which Data Set and Events Listing have been selected
When a Data Set is selected and no Events Listing has been selected, the time frame of the Data Set appears in the ***”Selected Time Frame***” fields. Selecting an Events Listing when no Data Set has been selected makes appear in the ***”Selected Time Frame***” fields the time frame of the Events Listing. Selecting both shows in the ***”Selected Time Frame***” fields the combination of both time frames.

Copyright per DIN 34

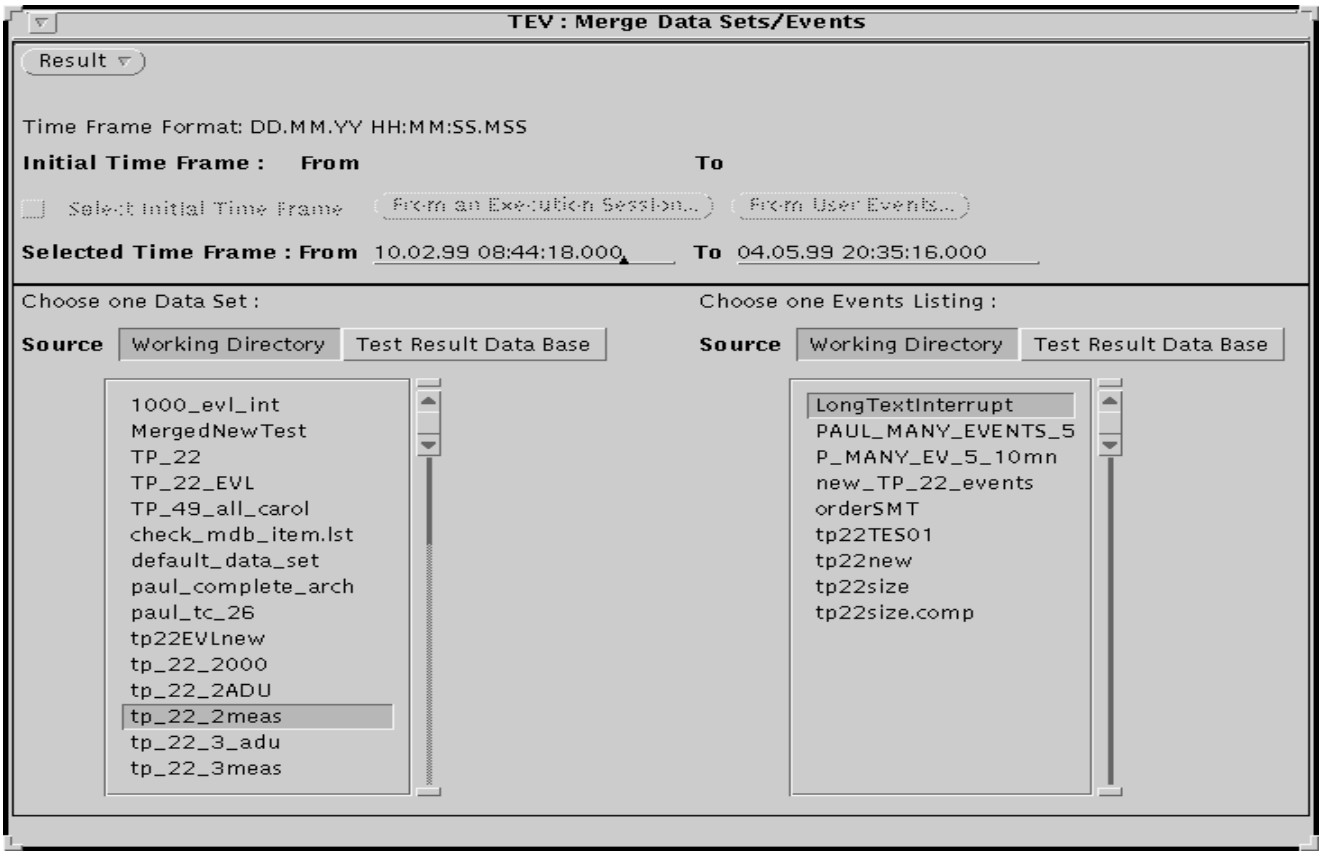


Figure 8–46 : *No Session Selected : Time Frame Combination*

When a session has been previously selected, the user can either use the time frame of the selected session (The ***"Initial Time Frame"***) or the combination of the time frames from a Data Set and an Events Listing. In this last case, the user shall first deselect the ***Initial Time Frame*** : deselect the Choice Box ***"Select Initial Tine Frame".*** If the Data Set and/or Events Listing were already selected before deselecting the ***Initial Time Frame,*** the Selected Time Frame is still setted–up to the ***Initial Time Frame.*** This allows the user to work on the base of the Session Time Frame. The user shall select again the Data Set and/or Events Listing in order to select explicitely the corresponding time frames.

Copyright per DIN 34

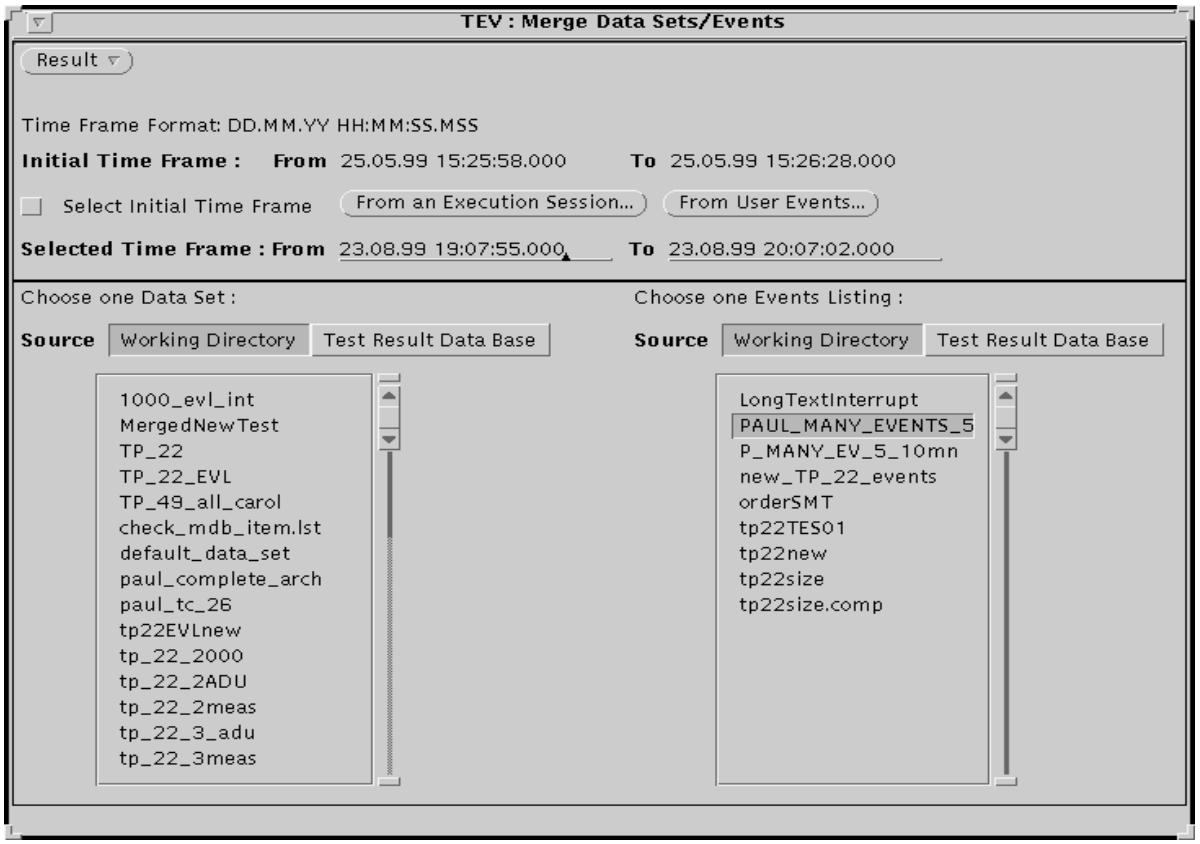


Figure 8–47 : *Session Selected with Selected Time Frame from Events Listing*

Once a Data Set and an Events Listing have been selected, the user starts the merge by clicking : ***Result:Exce&Display.***

The Data Set and the Events Listing shall have been generated with the same ***Order By*** option. Otherwise the error message ***"Error: Files to merge must be ordered by same time type"*** is displayed. The Result can be saved in the Working Directory using the menu option : ***"Result:Save"***.

To inspect a previously saved result, the user shall use the menu option : ***"Result:Load"***. The results of a Data Set/Events merging shall not be considered as a Data Sets : they can not be used for Statistics, Graph, or Data Set Listing. They can not be converted to the Excel format They can only be viewed through the Data Set / Events Merger Utilities.

Like for the Data Set Listing, the BYTE STREAM parameters are skipped (not shown) if some other types of parameters are included. If only BYTE STREAM parameters are included in the data set, then only the first one will be shown. The user has no means here to select the parameters to be shown. The Data Set shall already include the wanted parameters.

Copyright per DIN 34

9 TRDB TOOLS

9.1 General

The DBS application controlling the Test Result Data Base (TRDB) proposes one additional tool :

- The Recovery Scripts to be used to control and manage the consistency of the TRDB in cases of errors.

9.2 The Recovery Scripts

The Recovery Scripts allows the user to recover some inconsistencies appearing in the TRDB, due to DBS operation failures.

9.2.1 Getting Started

The Recovery Scripts are launched by invoking the C shell script *$DBS_HOME/util/common/recovery/recovery.csh* in the *DBS owner*'s user account.

The following menu is displayed :

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit

Enter your choice :

Figure 9–1 : *The Main Menu of the Recovery Scripts*

9.2.1.1 Execution Session

Select the Execution Session Menu by entering '1' in the Main Menu. The Execution Session Menu allows to call all the recovery scripts operations related to test execution sessions. It is shown in Figure 9–2. The

Copyright per DIN 34

user has to chose between 3 items. The choice 0 returns to the main menu.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit

Enter your choice : 1

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 1

1.1 Session Data Diagnostic

Enter session name :

Figure 9–2 : *Selection of the Execution Session Menu from the Main Menu*

9.2.1.1.1 Execution Session Diagnostic

As shown in the figure Figure 9–2, if the selection in the Execution Session Menu is ’1’, the user is asked for the session whose data diagnostic is required.

This functionality allows the scanning of the session data references and the display of a report. The information provided summarizes the session status, and helps the user to decide which recovery action is required. The Figure 9–3 shows such a report for a session named EXEC_BEN_005 which is partially retrieved from the Final Archive.

A report contains 8 sections. The 5 first sections list the references of the session data files. The division is based on their FILE_STATUS :

Copyright per DIN 34

- Central ACcessible (CAC) files: these are files normally stored into the TRDB, accessible for evaluation (’on–line’).
- Local ACcessible (LAC) files: these are files that local applications expect to be stored into the TRDB by Central DBS; they are not yet accessible for evaluation. This state is normally temporary during the storage into the TRDB by Central DBS (OPEN sessions). A file whose storage into the TRDB failed may remain in the state Local Accessible
- Central Not Accessible (CNA) files: these are files managed by Central DBS but not accessible for evaluation. It can be an error case.
- Local Not Accessible (LNA) files: these are files not accessible for evaluation and to Central DBS. It is an error case.
- ARChived (ARC) files: these are files archived on an Final Archive medium and not retrieved on the TRDB disk (not on–line).

The sixth section of a report contains a summary of the session Oracle Event table and its reference. If the Events of a session are on–line, an Oracle Event table must exist.

The seventh section lists the contents of the session WORK directory. This directory contains only temporary files. As soon as the session is closed, it should be empty. It may contain file whose storage into the TRDB failed.

The last section displays the session state. It checks the consistency of the session data state compared with the session state and summarize the situation.

The last report generated by the Recovery Scripts is maintained into the file ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***.

Recovery Execution Session Diagnostic : *<EXEC_BEN_005>*

The diagnostic analyses the TRDB references for session EXEC_BEN_005
i.e. the filename location, the session state/session data state.
Files are classified as :

– Central Accessible (normally stored into the TRDB, accessible for evaluation),
– Local Accessible (files that Local nodes expect to store into TRDB),
– Central Not Accessible (files that are not accessible for evaluation),
– Local Not Accessible (files that are not accessible for evaluation and to Central),
– Archived (files that are archived on an FA SAS Medium).

The latest diagnostic is kept into a file : ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***

1.1 Listing of Central Accessible files referenced in the TRDB

Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RAW_DATA/TEV_06/qwerty23
Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RESULT/qwerty233
Reference of File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/EVENT/
EVENT_TABLE04–03–96_09:31:28:616

It is a reference for an event file, file not found but event table must exist (see 1.6)

Listing of Central Accessible files finished :3 file(s)

1.2 Listing of Local Accessible files referenced in the TRDB

No Local Accessible file referenced in TRDB

Copyright per DIN 34

Listing of Local Accessible files finished :0 file(s)

1.3 Listing of Central NOT Accessible files referenced in the TRDB

No Central NOT Accessible file referenced in TRDB
Listing of NOT Accessible files finished :0 file(s)

1.4 Listing of Local NOT Accessible files referenced in the TRDB

No Local NOT Accessible file referenced in TRDB
Listing of NOT Local Accessible files finished :0 file(s)

1.5 Listing of Archived files referenced in the TRDB

Reference of File found : Final_Archive/EXECUTION/EXEC_BEN_005/ENG_VAL/TEV_06/TEV_06_04–03–96_09:29:22:558.EVL
Disk name is : MOD*ARCH*15.02.96–17:35
Listing of Archived files finished :1 file(s)

Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RAW_DATA/TEV_06/qwerty23
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/RESULT/qwerty233
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found : $DBS_HOME/data/EXECUTION/EXEC_BEN_005/EVENT/EVENT_TABLE04–03–96_09:31:28:616
(should match a Central Accessible Reference).
Disk name is : MOD*ARCH*15.02.96–17:35
Listing of Retrieved files finished :3 file(s)

1.6 Listing of Event Oracle Table referenced in the TRDB

TRDB does contain an Oracle Event Table for EXEC_BEN_005
Event Number : 1
Listing of Event Oracle Table finished

1.7 Listing of Session Work Directory

Session Work Directory does exist
path : $DBS_HOME/data/WORK/EXEC_BEN_005
content : Directory is empty
Listing of Session Work Directory finished

1.8 Checking of Session data

Session is recorded as ARCHIVED on FA Medium

Copyright per DIN 34

```
Data could be on FA Medium only (archived) or
        on FA Medium and on TRDB disk (retrieved from FA).
1 file(s) is(are) referenced as on FA Medium only
3 file(s) is(are) referenced as Central Accessible and Retrieved
Checking of Session data finished
```

Figure 9–3 : *Execution Session DataDiagnostic Report.*

9.2.1.1.2 Delete Execution Session Menu

As shown in the Figure 9–4, if the selection in the Execution Session Menu is ’2’, the Delete Execution Session Menu is displayed.

```
 – 1. Execution Session Menu –
 ____________________________

 1. Session Data Diagnostic
 2. Delete Session
 3. Session is Used
 4. File Storage Failure
 5. Close Session
 6. List Sessions
 7. File Transfer from Local Dir Failures
 8. Close aborted test sessions to NCL

 0. Exit

Enter your choice : 2

 1.2 Delete Session


 – 1.2 Delete Execution Session Menu –
 ____________________________________

 1. Delete Default Test Session Data (Session Refs remain)
 2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
 3. Delete Completely a Test Execution Session (Data and Refs)

 0. Exit

 Enter your choice :
```

Figure 9–4 : *Selection of the ’Delete Execution Session’ Menu*

This menu contains all the recovery scripts operations related to the deletion of test execution sessions. It displays 3 possible actions.

Copyright per DIN 34

- **Delete Default Test Session Data** (Session Refs remain) is selected by entering ’1’. This operation deletes all the data of the Default Test Session and leaves the session empty. The user is first asked to confirm the deletion request (see Figure 9–5). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the data of the Default Test Session, keeping the session references.

– 1.2 Delete Execution Session Menu –

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 1

1.2.1 Delete Default Test Session Data

This operation deletes all the data of the DEFAULT_TEST_SESSION.
The DEFAULT_TEST_SESSION remains open and empty.

CAUTION: This action will delete all the data of the DEFAULT TEST SESSION.
DEFAULT TEST SESSION will remain but empty.
Do you want to continue (y/n) :

Figure 9–5 : *Selection of the ’Delete Default Test Session Data’ Operation.*

- **Delete the On–Line Data of an Archived Test Execution Session** (Refs remain) is selected by entering ’2’. An example is given at the Figure 9–6 The user wants to delete the on–line data of the archived session SESSION_QWERTY. He is first asked for the name of the archived session concerned. Secondly, he has to confirm the deletion request (see Figure 9–6). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the on–line data of the given session and resets the session references accordingly. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’no match’, they can be ignored.

Copyright per DIN 34

– 1.2 Delete Execution Session Menu –

1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 2

1.2.2 Delete the On–Line Data of an Archived Test Execution Session (Refs remain)

This operation deletes the on–line data from the TRDB disk for a session which has been archived. It updates the Oracle references accordingly (session state ARCH, no owner, files ARC).

Enter session name (ONLY sessions that were archived): SESSION_QWERTY

CAUTION: This action will delete all the on–line data of SESSION_QWERTY. Be sure that the session data is archived on Final Archive (no automatic check). Do you want to continue (y/n) : ▮

Figure 9–6 : *Selection of 'Delete the On–Line Data of an Archived Session' Operation.*

- **Delete Completely a Test Execution Session (Data and Refs)** is selected by entering '3'. An example is given Figure 9–7. The user wants to delete completely the session SESSION_TEST. He is first asked for the name of the session concerned. Secondly, he has to confirm the deletion request (see Figure 9–7). If he enters 'y' or 'Y', the Recovery Scripts delete all the data and references of the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'no match', they can be ignored (see Figure 9–7).

Copyright per DIN 34

```
– 1.2 Delete Execution Session Menu –
_____________________________________


1. Delete Default Test Session Data (Session Refs remain)
2. Delete the On–Line Data of an Archived Test Execution Session (Refs remain)
3. Delete Completely a Test Execution Session (Data and Refs)

0. Exit

Enter your choice : 3

        1.2.3 Delete Completely a Test Execution Session (Data and Refs)

        Enter session name :  SESSION_TEST
        CAUTION: This action will delete all the data of    SESSION_TEST.
        Do you want to continue (y/n) :  y
        – deleting session Oracle data –
        – deleting session directories –
rm: $DBS_HOME/data/EXECUTION/SESSION_TEST/RAW_DATA: No such file or directory
        – listing participating applications –
        – removing list of participating applications –
```

Figure 9–7 : *Selection of the 'Delete Completely a Test Execution Session' Operation.*

9.2.1.1.3 Session Is Used Menu

As shown in the Figure 9–8, if the selection in the 'Execution Session Menu' is '3', the 'Session is Used Menu' is displayed.

The TRDB could remain in an inconsistent state if an operation related to the Final Archive (IMPORT and EXPORT) or the deletion of a session fails. This menu contains all the recovery actions to save the TRDB from such a failure. It displays 5 possible items.

- **Import failure** is selected by entering '1'.This operation deletes the data and references of a session whose import from Final Archive failed. After this operation has been executed, an application can request another Import of the same session.

  The first step is a deletion of the temporary data created for the IMPORT and EXPORT operations. In case no such data remain, an error message 'No match' could appear.

  The second step is the listing of all the sessions referenced as 'to be imported'. The user is asked to enter the name of a session out of this list. Finally, he is asked to confirm the deletion request (see Figure 9–8). If he enters 'y' or 'Y', the Recovery Scripts delete all the data of the given session.

Copyright per DIN 34

```
–  1.3 Session is Used Menu  –
_____________________________

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

1.3.1 Import Failure

        This operation deletes the temporary data of a session whose import has failed.

        a. Deletion of Import temporary directories
No match.
        b. Deletion of the session whose Import failed.
        The list of sessions recorded as "To Be Imported" is:
                Session EXEC_BEN_002 , Session state TBIM .

        Enter session name (a name from the list): EXEC_BEN_002

        CAUTION: This action will delete all the data of  EXEC_BEN_002.
        Do you want to continue (y/n) : 
```

Figure 9–8 : *Import from FA Failure, Execution Session.*

- **Export failure** is selected by entering '2'.

  This operation deletes the temporary data created for the Export operations. It updates also the session state to its initial value for the session whose export on Final Archive failed. After this recovery action, an application can request another export of the same session

  The first step is the deletion of the temporary data created for the Export operations. In case no such data remain, an error message 'No match' could appear (see Figure 9–9).

  The second step is the update of the session state for the session whose export on Final Archive failed. Its state has to be reset at its initial value (Note that this initial state is included in some of the DBS error messages related to Export failure). The Recovery Scripts display the listing of all the sessions referenced as 'to be exported'. The user is asked to enter the name of a session out of this list. If the name is valid, the session data diagnostic is launched (see Figure 9–9), and the Recovery Scripts propose an initial session state. Finally, the user has to enter the initial state of the session and is asked to confirm the update. If he enters 'y' or 'Y', the Recovery Scripts updates the state of the given session with the given value.

Copyright per DIN 34

1.3.2 Export Failure

This operation deletes the temporary data created for all the exports of sessions. It allows to reset the session state of a session whose export has failed.

a. Deletion of Export temporary directories
No match.
b. Resetting state of a session whose Export failed.
The list of sessions recorded as "To Be EXported" is:

Session EXEC_BEN_002 , Session state TBEX .
Session EXEC_BEN_003 , Session state TBEX .
Enter session name (a name from the list): EXEC_BEN_003
Running the Session Diagnostic to find initial STATE of the session:

...
...
1.8 Checking of Session data

...
...
Checking of Session data finished

The session contains some files referenced as archived and only on FA.
The session state should be ARCHIVED (ARCH).

Enter the session STATE wished (CLOS, ONLI or ARCH):
CLOS if session contains ONLY files referenced as Central Accessible and not on FA.
ONLI if session contains ONLY files referenced as retrieved from FA.
ARCH if session contains some files referenced as archived and only on FA.
The session state proposed is: ARCH
ARCH
CAUTION: This action will reset the STATE of EXEC_BEN_003 to ARCH.
Do you want to continue (y/n) : ▮

Figure 9–9 : *Export on FA Failure, Execution Session.*

- **Archiving failure** is selected by entering '3'.

  This operation analyses the status of a session whose archiving to Final Archive failed. It proposes a recovery scenario. After this recovery action, an application can request another archiving on the same session.

  An example is given at the Figure 9–10. The user wants to recover from a failure during the archiving of the session EXEC_BEN_003. The Recovery Scripts list the sessions recorded as 'To Be ARchived'. EXEC_BEN_003 is the only one. The user is prompted to enter a

Copyright per DIN 34

session name. The session data diagnostic is launched on the given session (EXEC_BEN_003) and the Recovery Scripts propose a corrective action based on the diagnostic. The user has to confirm before the corrective scenario is executed. In the example, the session EXEC_BEN_003 contains only files referenced on a Final Archive medium. Its state should be ARCHIVED.

1.3.3 Archiving Failure

This operation analyses the data of a session whose archiving failed, and proposes a recovery action.

The list of sessions recorded as "To Be ARchived" is:

Session EXEC_BEN_003 , Session state TBAR .
Enter the name of session to recover: EXEC_BEN_003
Running the Session Diagnostic to find a recovery scenario:

...
1.8 Checking of Session data

...
Checking of Session data finished

The session contains ONLY files referenced as archived and only on FA.
***Recovery scenario** : The session state is set to ARCHIVED (ARCH), the session data remaining on TRDB disk is deleted (it is also on FA). The data on FA is referenced.*

CAUTION: This scenario will reset the state of EXEC_BEN_003 to ARCH and delete the session data remaining on TRDB disk.
Do you want to continue (y/n) :

Figure 9–10 : *Archiving on FA Failure, Execution Session.*

- **Retrieving failure** is selected by entering '4'.

  This operation deletes the on–line data from the TRDB disk for a session whose retrieving from the Final Archive failed. The Oracle references are updated accordingly (data not on–line, on FA only). After this recovery action, an application can request another retrieving on the same session.

  An example is given Figure 9–11 The user wants to recover from a failure during the retrieving of data from the archived session EXEC_TEST. He is prompted to enter a session name (EXEC_TEST is entered). He has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'No match', they can be ignored.

Copyright per DIN 34

1.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for a session whose retrieving failed. It updates the Oracle references accordingly.

Enter session name: EXEC_TEST
CAUTION: This action will delete the on–line data from the TRDB disk for EXEC_TEST.
Do you want to continue (y/n) : y
– deleting the on–line data –
No match.
No match.
– resetting the Oracle references –

It is adviced to run the session diagnostic

Figure 9–11 : *Retrieving from FA Failure, Execution Session.*

- **Deletion failure** is selected by entering ’5’.

  This operation helps to recover from a deletion failure:
  – the deletion of the on–line data of an archived session (partial deletion) or
  – the complete deletion of all data and references of a closed session.
  It completes the required deletion.

  An example is given Figure 9–12 The user wants to recover from a failure during the partial deletion of the archived session EXEC_BEN_002. He is prompted to enter a session name (EXEC_BEN_002 is entered). He has to confirm the on–line data deletion. for the given archived session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored.

Copyright per DIN 34

1.3.5 Deletion Failure

This operation deletes the session data from the TRDB disk for a session whose deletion failed. The deletion can be complete (data and references) or partial (only data, for archived sessions) The list of sessions recorded as "To Be Deleted" is :

Session EXEC_BEN_002 , Session state TBPD .
Enter session name: EXEC_BEN_002
Do you want :
1. a partial deletion (deletion of on–line data, for archived session),
2. a complete deletion (on–line data and references)
1
CAUTION: This action will delete all the on–line data of EXEC_BEN_002.
Do you want to continue (y/n) : y
– deleting the on–line data –
No match

It is adviced to run the session diagnostic

Figure 9–12 : *Deletion Failure, Execution Session.*

9.2.1.1.4 File Storage Failure Menu

If the selection in the 'Execution Session Menu' is '4', the 'File Storage Failure' menu is displayed, as shown in Figure 9–13 This menu contains all the recovery operations related to failure during storage of execution session files.

Copyright per DIN 34

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 4

1.4 File Storage Failure

– 1.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice :

Figure 9–13 : *File Storage Failure, Execution Session.*

- **List Local Files Remaining Unstored** is selected by entering ’1’.This operation executes the session data diagnostic for a given session and display a summary of the local unstored files (Local Accessible and Local Not Accessible files).

  An example is given at the figure Figure 9–14 The user wants to list the local files for the session EXEC_BEN_002. He is asked to enter a session name (see Figure 9–14).

Copyright per DIN 34

```
– 1.4. File Storage Failure Menu –
_____________________________________

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 1

        1.4.1 List Files Remaining Unstored

        This operation calls the session data diagnostic to
        list files unstored.

        Enter session name : EXEC_BEN_002
        ____________________________________________________

        Unstored_files Execution Session Diagnostic : <EXEC_BEN_002>
...
        1.2 Listing of Local Accessible files referenced in the TRDB
        ____________________________________________________

                Reference of File found :  /ariane6/projects/VICOS/rt_user3/qwerty22
                Listing of Local Accessible files finished :1 file(s)
...
        1.8 Checking of Session data
        ____________________________

                Session is recorded as CLOSED
                Only Central Accessible files should exists.
                1 file(s) is(are) referenced as local accessible
                * These files and their references should be removed.
        Checking of Session data finished
        ____________________________________________________

        1  Local Accessible files have to be deleted
        0  Local Not Accessible files have to be deleted
```

Figure 9–14 : *List Local Files Unstored, Execution Session.*

- **Remove Local Files Remaining Unstored** is selected by entering ’2’. This operation deletes the Oracle references to the unstored files of a given session (Local Accessible and Local Not Accessible files).

  An example is given Figure 9–15 The user wants to recover from a failure during the storage of a file for session EXEC_BEN_002. The user is asked to enter a session name (see Figure 9–15) and can specify a file type. He has to confirm the deletion. **The flat files corresponding to the references deleted have to be removed manually from their location**.

Copyright per DIN 34

– 1.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 2

1.4.2 Remove Local Files Remaining Unstored

This operation removes the Oracle references to Local Files Remaining Unstored.
The flat files corresponding have to be deleted manually.

Enter session name : EXEC_002
Enter file type (RD / RESULT / EVENT / EVL / CONFIG / ALL) : RD
CAUTION: This action will delete the Oracle refs to RD Local Files of EXEC_002.
Do you want to continue (y/n) : ▮

Figure 9–15 : *Remove Local Files Unstored, Execution Session.*

9.2.1.1.5 Close Session

If the selection in the 'Execution Session Menu' is '5', the 'Close Session' operation is executed.

This operation allows the user to force the closure of a session whose normal closure failed. An example is given Figure 9–16 The user wants to recover from a failure during the closure of EXEC_BEN_007. It starts by the listing of all the open sessions. The user is asked to enter a session name out of the list and to confirm the action. If he enters 'y' or 'Y', the Recovery Scripts will:

- set the session state to CLOSED (CLOS),
- remove the references to participating applications to warn applications that the session is closed and
- set the temporary EVL files accessible.

Copyright per DIN 34

```
– 1. Execution Session Menu –
_____________________________

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit
Enter your choice : 5
        1.5 Close Session

        This operation forces the closure of a session whose normal closure failed.
        It sets the session state to CLOSED (CLOS), removes the references
        for the participating applications and sets temporary EVL files accessible.
        The list of sessions recorded as "OPEN" is:

              Session DEFAULT_TEST_SESSION , Session state OPEN .
              Session EXEC_BEN_007 , Session state OPEN .
              Session EXEC_BEN_008 , Session state OPEN .
        Enter session name (a name from the list): EXEC_BEN_007
        CAUTION: This action will force the closure  of  EXEC_BEN_007.
        Do you want to continue (y/n) : y
        – set session state to CLOS –
        – listing participating applications –
          TES_01 participating in session EXEC_BEN_008
                remains ...
          TES_07 participating in session EXEC_BEN_007
                removed ...
          TES_25 participating in session EXEC_BEN_007
                removed ...
        – removing list of participating applications –
        – set temporary EVL files accessible –
```

Figure 9–16 : *Close Execution Session.*

Copyright per DIN 34

9.2.1.1.6 List Execution Sessions

If the selection in the 'Execution Session Menu' is '6', the 'List Sessions' operation is executed.

This operation allows the user to list all the execution sessions recorded in the TRDB. The data displayed contain the session name, the session state, the test mode, the closure status, the session owner and the session creator (see Figure 9–17).

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 6

1.6 List Sessions

This operation lists all the execution sessions recorded in the TRDB. The data displayed is : the session name, the session state (OPEN, CLOSed, ARCHived, ONLIne), the test mode (1 character), the closure status (normally closed, aborted, open), the session owner and the session creator.

NAME	STAT	T	CLO	OWNER	CREATOR
IMPORTED	CLOS	A	NCL	rt_user3	rt_user3
AFTER_IMPORT	CLOS	A	NCL	rt_user3	rt_user3
TEST_ARC1	ARCH	A	NCL		rt_user3
TEST_ARC2	ARCH	A	NCL		rt_user3
TEST_ARC3	ARCH	A	NCL		rt_user3
TEST_ARC4	ARCH	A	NCL		rt_user3
TEST_EVL	CLOS	A	ABT	rt_user2	rt_user2
DEFAULT_TEST_SESSION	OPEN	N	OPN		
EXEC_002	ARCH	A	NCL	rt_user7	rt_user3
EXEC_003	ARCH	A	NCL		rt_user3
EXEC_004	CLOS	A	NCL	rt_user3	rt_user3
EXEC_005	ARCH	A	NCL	rt_user3	rt_user3

12 rows selected.

Figure 9–17 : *Execution Session List.*

Copyright per DIN 34

9.2.1.1.7 File Transfer From Local Dir Failures

If the selection in the 'Execution Session Menu' is '7', a menu is provided which allows to handle failures in transferring archive or event files from local disks to the central TRDB's disks.

Having selected the test session (menu entry 1), the current files of a session may be listed or the local files may be transfered to the TRDB's central disk. Each node (testnode, workstation) is to be handled separately. The location of the local files have to be specified by the user.

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close aborted test sessions to NCL

0. Exit

Enter your choice : 7

1.7 File Transfer from Local Dir Failures

Recover from incomplete/not executed file transfers from local
DBS/archive directories (on test nodes or workstations)
Recovers EVT, EVL or archive files and adds them to a test session)

Recover Session Files Menu –

1. Recover files for multiple sessions (auto detection)
2. Specify Test Session
3. Recover all archive files (each node separately)
4. Recover event/evl files (each node separately)
5. List all files in Test Session
6. List Contents of Archive Files in Test Session (TOC: Pathnames)
7. List Items of Archive Files in Test Session (all archived entries)

0. Exit

Enter your choice :

Figure 9–18 : *File Transfer Failures.*

Copyright per DIN 34

Recover files for multiple sessions (auto detection) (Selection: 1)

This selection allows for recovery of files with ”auto detection” of the related test session, i.e. according to the creation date of the found files they will automatically be assigned to the correct testsession and be stored in the scope of that session. The following submenu is displayed:

```
Enter Testnode directory..... 1
Analyze only..... 2 [ YES ]
Define Logfile..... 3
Delete files on testnode after restore..... 4 [ YES ]
Assign unrelated files to specific session.. 5
Continue in case of error..... 6 [ NO ]
Analyze files..... 7
List sessions..... 8
Exit..... 9

Enter number of command:
```

Figure 9–19 : *Recover Files for Multiple Sessions with Auto–Detection*

Command description:

1. Enter Testnode directory
   The line ”Enter eventfile directory:” is displayed. The name of the testnode directory is expected here, e.g. /testnode/cgs–test. This name will be displayed below menu item [1]:
   ```
   Enter Testnode directory..... 1
     [/testnode/cgs–test]
   Analyze only..... 2 [ YES ]
   ```
2. Analyze only
   This command will have an influence on item [7], Analyze/Restore Files. If selected the value ”YES” displayed at the end of the line will switch to ”NO” and the line for menu item [7] will change from ”Analyze files” to ”Restore files” (see below).
3. Define Logfile
   If the question ”Store messages in file (y/n)? [n]” is answered with ”y” a logfile can be entered to contain all messages displayed in the scope of this recovery. If the question to enter the name of the logfile is simply answered with a <return> a default logfile named ”STORE_FILES.LOG” will be created in the current directory. The selected name will be displayed below menu item [3]:
   ```
   Define Logfile..... 3
     [STORE_FILES.LOG]
   Delete files on testnode after restore..... 4 [ YES ]
   ```
4. Delete files on testnode after restore
   On selection of this item the value ”YES” at the end of the line switches to ”NO”, i.e. the files will remain on the selected testnode directory after recovery.
5. Assign unrelated files to specific session
   If a file cannot be assigned to a testsession it may be related to a specific session. On selection of this item the name of that session can be entered:
   ```
   Add unrelated files to session (DEFAULT_TEST_SESSION):
   ```
   (default: DEFAULT_TEST_SESSION, selected on <return>)

Copyright per DIN 34

The selected name will be displayed below menu item [5]:

Assign unrelated files to specific session..... 5
[DEFAULT_TEST_SESSION]
Continue in case of error..... 6 [NO]

Note that files which cannot be assigned to a session will remain on the local directory in case no session has been selected alternatively.

6. Continue in case of error
   In case of any error during the recovery process the activity is stopped immediately. This selection will force the system to continue in case of errors.

7. Analyze files/Restore files
   In case the selection of item [2], Analyze only is set to ”YES” the files found on the specified testnode directory will be analyzed, i.e. the name of the found files is displayed together with the related testsession. If no session can be assigned this will be displayed accordingly:

   Eventfiles:
   /testnode/cgs–test/TSCV_01.28–03–00_10:26:48:597.EVT.002...assigned to session:TEST_01
   /testnode/cgs–test/HCI_01.28–03–00_10:30:34:680.EVT.007...assigned to session:TEST_02
   EVL files:
   /testnode/cgs–test/TSCV_01.28–03–00_10:26:48:619.EVL.003...assigned to session:TEST_01
   /testnode/cgs–test/HCI_01.28–03–00_10:29:49:531.EVL.003...not related to any session

   In case the selection of item [2], Analyze only is set to ”NO” the files found on the specified testnode directory will be stored in the TRDB in the scope of the assigned session.

8. List sessions
   This will display a list of all sessions.

9. Exit
   This selection will end the recovery session and will bring the user back to the previous menu.

Copyright per DIN 34

Specify Test Session (Selection: 2)

The name of the test session which is to be recovered is to be entered. It is applicable for all subsequent selections.

An existing execution session must be specified. The session may be in any state.

Recover all archive files (Selection: 3)

For each node, the test node's local directory must be specified (without subdirectory "archive")

After succesful transfer, the files should be deleted in the local directory.

```
Enter your choice : 2
Enter Test Node directory ([/testnode/host_name]): /testnode/cgs–test
–rw–r—r—  1 cgsadmin cgs      11264 May  7 18:18 TES_01001_199903111753.arc
–rw–r—r—  1 cgsadmin cgs     113664 May  7 18:18 TES_01002_199903111754.arc

Shall all of these files be added to the test session PAUL_RECOVERED  (y/[n]) ? y

...

Check that the files have been registered and copied to the test session
Shall they all be deleted now on the local disk ? (y/[n]) y
```

Figure 9–20 : *Recover Archive Files for one Test Node*

Recover event/evl files (Selection: 4)

For each test node, the node's local directory must be specified
For each workstation, the DBS work directory must be specified.

```
Enter your choice: 3
Default Location: /gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION
On test nodes: /testnode/<host_name>

Enter event/evl file directory to be added:

/gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION

No match

Note: EVL files will be deleted directly after insertion
Shall all of these files be added to the test session PAUL_RECOVERED  (y/[n]) ?n

cleanup directory /gsaf_home/dbs/data/WORK/DEFAULT_TEST_SESSION and start again
```

Figure 9–21 : *Recover Event/Evl Files for one Test Node*

List all files in test session (Selection: 5)

Copyright per DIN 34

The files already transfered and stored currently on the DB Server's central disk are listed.

List contents of archive files in test session / Pathames (Selection: 6)

The archive files already stored within the test session are scanned for items stored. A list is generated for each archive item type (SMT Updates, GDUs, ADUs) and for all types together.
The list shows each SID and pathname stored (at least once) in the files.
The lists are generated as ASCII files and stored under <test_session>.<type> resp. <test_session>.lst
They are presented to the user via the OpenLook Texteditor.

List contents of archive files in test session / Entries (Selection: 7)

The archive files already stored within the test session are scanned for items stored. A list of all items together with their timetag is generated for each archive item type (SMT Updates, GDUs, ADUs) and for all types together.
The lists are generated as ASCII files and stored under <test_session>.<type> resp. <test_session>.lst
They are presented to the user via the OpenLook Texteditor.

Copyright per DIN 34

9.2.1.1.8 Close Aborted Test Session to Normally Closed

If the selection in the 'Execution Session Menu' is '8', a function is provided which allows to close an aborted test session to the state NORMALLY CLOSED, thus allowing to export the session and to evaluate the session as usual.

Note: This function should be called **carefully:** The contents of the session might be corrupted! When setting the session state to NORMALLY CLOSED, no indication is availalbe anymore that something has gone wrong during execution of the session.

– 1. Execution Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Close Session
6. List Sessions
7. File Transfer from Local Dir Failures
8. Close Aborted Test Sessions to NCL

0. Exit

Enter your choice : 8

1.8 Close Aborted Test Sessions to Normally Closed

Allow for evaluation/final archiving of aborted test sessions
by setting it to normally closed
Should only be called for test sessions which cannot be handled otherwise

>>> Warning: The test session will appear as normally closed session <<<
>>> Even if there are inconsistencies of any kind within the session <<<

<list of sessions>

Enter session name (a name from the list): <SESSION>

CAUTION: This action will force the normal closure of <SESSION>
The session status will not indicate anymore any problem during closure
The session cannot be set to ABORTED again

Do you want to continue (y/n) : y

Figure 9–22 : *Set Aborted Session to Normally Closed*

Copyright per DIN 34

9.2.1.2 Evaluation Session Menu

Select the Evaluation Session Menu by entering ’2’ in the Main Menu. The Evaluation Session Menu allows to call all the recovery scripts operations related to test evaluation sessions. It is shown in figure Figure 9–23 The user has to chose between 5 items. The choice 0 returns to the Main Menu.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit

Enter your choice : 2

– 2. Evaluation Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 1

2.1 Session Data Diagnostic

Enter session name :

Figure 9–23 : *Selection of the Evaluation Session Menu from the Main Menu*

9.2.1.2.1 Evaluation Session Diagnostic

As shown in the Figure 9–23, if the selection in the Evaluation Session Menu is ’1’, the user is asked for the session whose data diagnostic is required.

This functionality allows the scanning of the session data references and the display of a report. The information provided summarizes the session status, and helps the user to decide which recovery action is required. The Figure 9–24 shows such a report for a session named EVAL_01 which is retrieved from the Final Archive.

A report contains 6 sections. The 5 first sections list the references of the session data files. The division is based on their FILE_STATUS :

Copyright per DIN 34

- Central ACcessible (CAC) files: these are files normally stored into the TRDB, accessible for evaluation (’on–line’).
- Local ACcessible (LAC) files: these are files that local applications expect to be stored into the TRDB by Central DBS; they are not yet accessible for evaluation. This state is normally temporary during the storage into the TRDB by Central DBS (OPEN sessions). A file whose storage into the TRDB failed may remain in the state Local Accessible
- Central Not Accessible (CNA) files: these are files managed by Central DBS but not accessible for evaluation. It can be an error case.
- Local Not Accessible (LNA) files: these are files not accessible for evaluation and to Central DBS. It is an error case.
- ARChived (ARC) files: these are files archived on an Final Archive medium and not retrieved on the TRDB disk (not on–line).

The last section displays the session state. It checks the consistency of the session data state compared with the session state and summarize the situation.

The last report generated by the Recovery Scripts is maintained into the file ***$DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic***.

Recovery Evaluation Session Diagnostic : <EVAL_01>

The diagnostic analyses the TRDB references for session EVAL_01
i.e. the filename location, the session state/session data state.
Files are classified as :

– Central Accessible (normally stored into the TRDB, accessible for evaluation),
– Local Accessible (files that Local nodes expect to store into TRDB),
– Central Not Accessible (files that are not accessible for evaluation),
– Local Not Accessible (files that are not accessible for evaluation and to Central),
– Archived (files that are archived on an FA SAS Medium).

The latest diagnostic is kept into a file: $DBS_HOME/util/sun5/recovery/work/Recovery.diagnostic

1.1 Listing of Central Accessible files referenced in the TRDB

Reference of File found: $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/GRAPH/q2
Reference of File found: $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/REPORT/qq
Listing of Central Accessible files finished :2 file(s)

1.2 Listing of Local Accessible files referenced in the TRDB

No Local Accessible file referenced in TRDB
Listing of Local Accessible files finished :0 file(s)

1.3 Listing of Central NOT Accessible files referenced in the TRDB

No Central NOT Accessible file referenced in TRDB
Listing of NOT Accessible files finished :0 file(s)

1.4 Listing of Local NOT Accessible files referenced in the TRDB

Copyright per DIN 34

```
No Local NOT Accessible file referenced in TRDB
Listing of NOT Local Accessible files finished :0 file(s)

1.5 Listing of Archived files referenced in the TRDB
-----------------------------------------------------------------------

No Archived file referenced in TRDB
Listing of Archived files finished :0 file(s)

Reference of Retrieved File found:  $DBS_HOME/data/EVALUATION/ EVAL_01/RESULT/GRAPH/q2
      (should match a Central Accessible Reference).
            Disk name is :  MOD*ARCH*15.02.96–17:35
Reference of Retrieved File found:  $DBS_HOME/data/EVALUATION/EVAL_01/RESULT/REPORT/qq
      (should match a Central Accessible Reference).
            Disk name is :  MOD*ARCH*15.02.96–17:35
Listing of Retrieved files finished :2 file(s)

1.6 Checking of Session data
-----------------------------------------------------------------------

Session is recorded as ON–LINE
All data should be Central Accessible, retrieved from FA.
2 file(s) is(are) referenced as retrieved from FA and Central Accessible
Checking of Session data finished
-----------------------------------------------------------------------
```

Figure 9–24 : *Report of the Evaluation Session Data Diagnostic*

9.2.1.2.2 Delete Evaluation Session Menu

If the selection in the Evaluation Session Menu is ’2’, the Delete Evaluation Session Menu is displayed.

This menu contains all the recovery scripts operations related to the deletion of evaluation sessions. It displays 2 possible actions.

- **Delete the On–Line Data of an Archived Evaluation Session** (Refs remain) is selected by entering ’1’. An example is given Figure 9–25 The user wants to recover from a failure during the partial deletion of EVAL_001. He is first asked for the name of the archived session concerned. Secondly, he has to confirm the deletion request (see Figure 9–25). If he enters ’y’ or ’Y’, the Recovery Scripts delete all the on–line data of the given session and resets the session references accordingly. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’no match’, they can be ignored.

Copyright per DIN 34

– 2.2 Delete Evaluation Session Menu –

1. Delete the On–Line Data of an Archived Evaluation Session (Refs remain)
2. Delete Completely an Evaluation Session (Data and Refs)

0. Exit

Enter your choice : 1

2.2.1 Delete the On–Line Data of an Archived Evaluation Session (Refs remain)

This operation deletes the on–line data from the TRDB disk for a session which has been archived. It updates the Oracle references accordingly (session state ARCH, no owner, files ARC).

Enter session name (ONLY sessions that were archived): EVAL_001
CAUTION: This action will delete all the on–line data of EVAL_001.
Be sure that the session data is archived on Final Archive (no automatic check).
Do you want to continue (y/n) : y
– deleting the on–line data –
– resetting the Oracle references –

It is adviced to run the session diagnostic

Figure 9–25 : *Selection of 'Delete the On–Line Data of an Archived Session', Evaluation Session.*

- **Delete Completely an Evaluation Session (Data and Refs)** is selected by entering '2'. An example is given Figure 9–26 The user wants to recover from a failure during the complete deletion of EVAL_004. He is first asked for the name of the session concerned. Secondly, he has to confirm the deletion request (see Figure 9–26). If he enters 'y' or 'Y', the Recovery Scripts delete all the data and references of the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'no match', they can be ignored.

Copyright per DIN 34

– 2.2 Delete Evaluation Session Menu –

1. Delete the On–Line Data of an Archived Evaluation Session (Refs remain)
2. Delete Completely an Evaluation Session (Data and Refs)

0. Exit

Enter your choice : 2

2.2.2 Delete Completely an Evaluation Session (Data and Refs)

Enter session name : EVAL_004
CAUTION: This action will delete all the data of EVAL_004.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –

Figure 9–26 : *Selection of 'Delete Completely an Evaluation Session' Operation.*

9.2.1.2.3 Session Is Used Menu

If the selection in the 'Evaluation Session Menu' is '3', the 'Session is Used Menu' is displayed.

The TRDB could remain in an inconsistent state if an operation related to the Final Archive (ARCHIVING, RETRIEVING, IMPORT and EXPORT) or the deletion of a session fails. This menu contains all the recovery actions to recover the TRDB from such a failure. It displays 5 possible items.

- **Import failure** is selected by entering '1'.This operation deletes the data and references of a session whose import from Final Archive failed. After this operation has been executed, an application can request another Import of the same session.

  The first step is a deletion of the temporary data created for the IMPORT and EXPORT operations. In case no such data remain, an error message 'No match' could appear. It can be ignored.

  The second step is the listing of all the evaluation sessions referenced as 'to be imported'. The user is asked to enter the name of a session out of this list. Finally, he is asked to confirm the deletion request (see Figure 9–27). If he enters 'y' or 'Y', the Recovery Scripts delete all the data of the given session.

Copyright per DIN 34

```
–    2.3 Session is Used Menu    –
_____________________________________

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

        2.3.1 Import Failure

        This operation deletes the temporary data of a session whose
        import has failed.

        a. Deletion of Import temporary directories
No match.
        b. Deletion of the session whose Import failed.
          The list of sessions recorded as "To Be Imported" is:

                Session EVAL_004 , Session state TBIM .
        Enter session name (a name from the list): EVAL_004
        CAUTION: This action will delete all the data of  EVAL_004.
        Do you want to continue (y/n) : y
                – deleting session Oracle data –
                 – deleting session directories –
        _____________________________________________

                It is adviced to run the session diagnostic
        _____________________________________________
```

Figure 9–27 : *Import from FA Failure, Evaluation Session.*

- **Export failure** is selected by entering '2'.

  This operation deletes the temporary data created for the Export operations. It updates also the session state to its initial value for the session whose export on Final Archive failed. After this recovery action, an application can request another export of the same session

  The first step is the deletion of the temporary data created for the Export operations. In case no such data remain, an error message 'No match' could appear (see Figure 9–28).

  The second step is the update of the session state for the session whose export on Final Archive failed. Its state has to be reset at its initial value (Note that this initial state is included in some of the DBS error messages related to Export failure). The Recovery Scripts display the listing of all the sessions referenced as 'to be exported'. The user is asked to enter the name of a session out of this list. If the name is valid, the session data diagnostic is launched

Copyright per DIN 34

(see Figure 9–28), and the Recovery Scripts propose an initial session state. Finally, the user has to enter the initial state of the session and is asked to confirm the update. If he enters 'y' or 'Y', the Recovery Scripts updates the state of the given session with the given value.

2.3.2 Export Failure

This operation deletes the temporary data created for all the exports of sessions. It allows to reset the session state of a session whose export has failed.

a. Deletion of Export temporary directories
No match.
b. Resetting state of a session whose Export failed.
The list of sessions recorded as "To Be EXported" is:

Session EVAL_002 , Session state TBEX .
Enter session name (a name from the list): EVAL_002
Running the Session Diagnostic to find initial STATE of the session:

Session_is_Used Evaluation Session Diagnostic : <EVAL_002>
...
1.6 Checking of Session data

...
Checking of Session data finished

The session contains ONLY files referenced as Central Accessible and not on FA.
The session state should be CREATED (CREA).

Enter the session STATE wished (CREA, ONLI or ARCH):
CREA if session contains ONLY files referenced as Central Accessible and not on FA.
ONLI if session contains ONLY files referenced as retrieved from FA.
ARCH if session contains ONLY files referenced as archived and only on FA.
The session state proposed is: CREA
CREA
CAUTION: This action will reset the STATE of EVAL_BEN_002 to CREA.
Do you want to continue (y/n) : y
– resetting session state –

It is adviced to run the session diagnostic

Figure 9–28 : *Export on FA Failure, Evaluation Session.*

- **Archiving failure** is selected by entering '3'.

  This operation analyses the status of a session whose archiving to Final Archive failed. It proposes a recovery scenario. After this recovery action, an application can request another

Copyright per DIN 34

archiving on the same session.

An example is given Figure 9–29 The user wants to recover from a failure during the archiving of EVAL_002 on FA. The Recovery Scripts list the sessions recorded as 'To Be ARchived'. EVAL_002 is the only one. The user is prompted to enter a session name. The session data diagnostic is launched on the given session (EVAL_002) and the Recovery Scripts propose a corrective action based on the diagnostic. The user has to confirm before the corrective scenario is executed. In the example, the session EVAL_002 contains files referenced only on the TRDB disk. Its state should be CREATED (CREA).

2.3.3 Archiving Failure

This operation analyses the data of a session whose archiving failed, and proposes a recovery action.

The list of sessions recorded as "To Be ARchived" is:

Session EVAL_002 , Session state TBAR .
Enter the name of session to recover: EVAL_002
Running the Session Diagnostic to find a recovery scenario:

Session_is_Used Evaluation Session Diagnostic : <EVAL_002>

...

1.6 Checking of Session data

Session is recorded as To Be Archived

...

Checking of Session data finished

The session contains ONLY files referenced as Central Accessible and not on FA.
These files are all accessible.
Recovery scenario : The session state is set to CREATED (CREA).

CAUTION: This scenario will reset the state of EVAL_002 to CREATED.
Do you want to continue (y/n) : ▮

Figure 9–29 : *Archiving on FA Failure, Evaluation Session.*

- **Retrieving failure** is selected by entering '4'.

This operation deletes the on–line data from the TRDB disk for a session whose retrieving from the Final Archive failed. The Oracle references are updated accordingly (data not on–line, on FA only). For a retrieving with a NEW_NAME, the **Import Failure** operation (see 9.2.1.2.3.) must be called afterwards to complete the corrective actions. After this, an application can request another retrieving on the same session.

Two examples are given at the figure Figure 9–30 and Figure 9–31 In the first one, the user wants to recover from a failure during the retrieving of EVAL_001 with a null

Copyright per DIN 34

NEW_NAME. The user is prompted to enter a session name (EVAL_001 is entered). He has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'No match', they can be ignored.

In the second example (see Figure 9–31), the user wants to recover from a failure during the retrieving of EVAL_001 with NEW_NAME = EVAL_002. He has called the 'Session is Used' menu, and has chosen the fourth option 'Retrieving Failure'. The Recovery Scripts ask for a session name (EVAL_001 is entered). The user has to confirm the on–line data deletion for the given session. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern 'No such file of directory' or 'No match', they can be ignored. After completion of the operation, the menu 'Session is Used' is displayed. The user selects the first option 'Import Failure' and deletes the temporary data of the session EVAL_002.

2.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for a session whose retrieving failed. It updates the Oracle references accordingly.
For a retrieving with a NEW_NAME, the Menu <Import Failure> must be used afterwards to delete the NEW_NAME session.

Enter session name: EVAL_BEN_001
CAUTION: This action will delete the on–line data from TRDB disk for EVAL_001
Do you want to continue (y/n) : y
– deleting the on–line data –
– resetting the Oracle references –

It is adviced to run the session diagnostic

Figure 9–30 : *Retrieving from FA Failure, Evaluation Session, no NEW_NAME.*

Copyright per DIN 34

2.3.4 Retrieving Failure

This operation deletes the on–line data from the TRDB disk for a session whose retrieving failed. It updates the Oracle references accordingly.
For a retrieving with a NEW_NAME, the Menu <Import Failure> must be used afterwards to delete the NEW_NAME session.

Enter session name: EVAL_BEN_001
CAUTION: This action will delete the on–line data from TRDB disk for EVAL_001
Do you want to continue (y/n) : y
– deleting the on–line data –
– resetting the Oracle references –

It is adviced to run the session diagnostic

– 2.3 Session is Used Menu –

---

1. Import Failure
2. Export Failure
3. Archiving Failure
4. Retrieving Failure
5. Deletion Failure

0. Exit

Enter your choice : 1

2.3.1 Import Failure

This operation deletes the temporary data of a session whose import has failed.
a. Deletion of Import temporary directories
No match.
b. Deletion of the session whose Import failed.
The list of sessions recorded as "To Be Imported" is:

Session EVAL_002 , Session state TBIM .
Enter session name (a name from the list): EVAL_002
CAUTION: This action will delete all the data of EVAL_002.
Do you want to continue (y/n) : y
– deleting session Oracle data –
– deleting session directories –

It is adviced to run the session diagnostic

Copyright per DIN 34

Figure 9–31 : *Retrieving from FA Failure, Evaluation Session, NEW_NAME not null.*

- **Deletion failure** is selected by entering ’5’.

  This operation helps to recover from a deletion failure:
  – the deletion of the on–line data of an archived session (partial deletion) or
  – the complete deletion of all data and references of a closed session.
  It completes the required deletion.

  An example is given Figure 9–32 The user wants to recover from a failure in the complete deletion of EVAL_002. He is prompted to enter a session name (EVAL_002 is entered). He has to confirm the session deletion. Some error messages could appear while deleting the session directories if the session does not contain data of all types. As far as they contain the pattern ’No such file of directory’ or ’No match’, they can be ignored.

```
        2.3.5 Deletion Failure

        This operation deletes the session data from the TRDB disk for
        a session whose deletion failed.
        The list of sessions recorded as "To Be Deleted" is :

                Session EVAL_002 , Session state TBCD .
        Enter session name: EVAL_002
        Do you want :
        1. a partial deletion (deletion of on–line data, for archived session),
        2. a complete deletion (on–line data and references)
2
        CAUTION: This action will delete all the data of  EVAL_BEN_002.
        Do you want to continue (y/n) : y
  – deleting session Oracle data –
  – deleting session directories –
```

Figure 9–32 : *Deletion Failure, Evaluation Session.*

9.2.1.2.4 File Storage Failure Menu

If the selection in the ’Evaluation Session Menu’ is ’4’, the ’File Storage Failure’ menu is displayed, as shown in Figure 9–33. This menu contains all the recovery operations related to failure during storage of evaluation session files.

Copyright per DIN 34

```
– 2. Evaluation Session Menu –
_____

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 4

2.4 File Storage Failure

        – 2.4. File Storage Failure Menu –
        _____

        1. List Local Files Remaining Unstored
        2. Remove Local Files Remaining Unstored

        0. Exit

        Enter your choice : 
```

Figure 9–33 : *File Storage Failure, Evaluation Session.*

It displays 2 possible items.

- **List Local Files Remaining Unstored** is selected by entering ’1’.This operation executes the session data diagnostic for a given session and display a summary of the local unstored files (Local Accessible and Local Not Accessible files).

  An example is given at the figure Figure 9–34 The user wants to list the local files for the session EVAL_002. He is asked to enter a session name (see Figure 9–34).

Copyright per DIN 34

– 2.4. File Storage Failure Menu –

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 1

2.4.1 List Files Remaining Unstored

This operation calls the session data diagnostic to list files unstored.

Enter session name : EVAL_002

Unstored_files Evaluation Session Diagnostic : <EVAL_002>

...

1.2 Listing of Local Accessible files referenced in the TRDB

Reference of File found : /ariane6/projects/VICOS/rt_user3/qwerty22
Listing of Local Accessible files finished :1 file(s)

...

1.8 Checking of Session data

Session is recorded as CREATED (on–line and not archived)
1 file(s) is(are) referenced as local accessible
** Note that at least one file has not been found at its referenced location.*
Checking of Session data finished

1 Local Accessible files have to be deleted
0 Local Not Accessible files have to be deleted

Figure 9–34 : *List Local Files Unstored, Evaluation Session.*

- **Remove Local Files Remaining Unstored** is selected by entering ’2’. This operation deletes the Oracle references to the unstored files of a given session (Local Accessible and Local Not Accessible files).

  An example is given Figure 9–35. The user wants to recover from a failure during the storage of a file for session EVAL_002. The user is asked to enter a session name (see Figure 9–35) and to confirm the deletion. **The flat files corresponding to the references deleted have to be removed manually from their location**.

Copyright per DIN 34

```
– 2.4. File Storage Failure Menu –
_____________________________

1. List Local Files Remaining Unstored
2. Remove Local Files Remaining Unstored

0. Exit

Enter your choice : 2

        2.4.2 Remove Local Files Remaining Unstored

        This operation removes the Oracle references to Local Files Remaining Unstored.
        The flat files corresponding have to be deleted manually.

        Enter session name : EVAL_002
        CAUTION: This action will delete the Oracle refs to Local Files of  EVAL_002.
        Do you want to continue (y/n) : 
```

Figure 9–35 : *Remove Local Files Unstored, Evaluation Session.*

9.2.1.2.5 Remove Evaluation Users Menu

If the selection in the ’Evaluation Session Menu’ is ’5’, the ’Remove Evaluation Users’ menu is selected. This menu (see Figure 9–36) provides the operations related to evaluation user references, i.e.

- a reference for the evaluation connection,
- references of sessions allocated for evaluation.

Copyright per DIN 34

```
– 2. Evaluation Session Menu –
_____________________________

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 5

        – 2.5 Remove Evaluation Users –
        _____________________________

        1. List Connection References
        2. Delete Connection References
        3. List Allocation References
        4. Delete Allocation References

        0. Exit

        Enter your choice : 
```

Figure 9–36 : *Remove Evaluation Users Menu.*

The menu shows 4 items.

- **List Connection References** is selected by entering ’1’.This operation displays the applications currently recorded as connected for evaluation purposes.

  An example is given Figure 9–37 There are two evaluation applications currently recorded in the TRDB, TEV_07 and TEV_01.

Copyright per DIN 34

– 2.5 Remove Evaluation Users –

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 1

2.5.1 List Connection References

This operation lists the applications referenced as connected for evaluation purposes.

Evaluation_Application

TEV_07
TEV_01

Figure 9–37 : *List the Connection References of Evaluation Applications*

- **Delete Connection References** is selected by entering ’2’.This operation allows to delete the connection reference of an evaluation application that crashed.

  An example is given Figure 9–38. The user is asked for the name of the application whose reference must be deleted. He must also confirm the deletion.

Copyright per DIN 34

```
– 2.5 Remove Evaluation Users –
_____

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 2

2.5.2 Delete Connection References

This operation deletes the connection reference
for the given application.

Enter the application name to be removed : TEV_07

CAUTION: This operation will delete the connection reference of  TEV_07
Do you want to continue (y/n) : 
```

Figure 9–38 : *Delete the Connection References of an Evaluation Application*

- **List Allocation References** is selected by entering ’3’.This operation displays the allocations of sessions currently recorded in the TRDB for all the evaluation users.

  An example is given Figure 9–39 The information provided is the evaluation user name, the sessions that he has allocated and their type (EX for EXecution and EV for EValuation).

Copyright per DIN 34

– 2.5 Remove Evaluation Users –

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 3

2.5.3 List Allocation References

This operation lists the session allocations currently recorded in the TRDB.

Evaluation_User	*SESSION_NAME*	*TY*
rt_user3	*EXEC_BEN_002*	*EX*
rt_user3	*EXEC_BEN_011*	*EX*
rt_user3	*EXEC_BEN_006*	*EX*
rt_user2	*EVAL_001*	*EV*
rt_user1	*EVAL_001*	*EV*

Figure 9–39 : *List the Allocation References of all Evaluation Users*

- **Delete Allocation References** is selected by entering ’4’.This operation allows to delete the allocation references of a given evaluation user.

  An example is given Figure 9–40. The user is asked for the name of the user whose allocations must be deleted. He must also confirm the deletion.

Copyright per DIN 34

```
– 2.5 Remove Evaluation Users –
_____

1. List Connection References
2. Delete Connection References
3. List Allocation References
4. Delete Allocation References

0. Exit

Enter your choice : 4

2.5.4 Delete Allocation References

Enter the User whose allocations must be removed : test_user23

CAUTION: This operation will delete the allocation reference of  test_user23
Do you want to continue (y/n) : 
```

Figure 9–40 : *Delete the Allocations of an Evaluation User*

9.2.1.2.6 List Evaluation Sessions

If the selection in the ’Evaluation Session Menu’ is ’6’, the ’List Sessions’ operation is executed.

This operation allows the user to list all the evaluation sessions recorded in the TRDB. The data displayed contain the session name, the session state, the session owner and the session creator (see Figure 9–41).

Copyright per DIN 34

– 2. Evaluation Session Menu –

1. Session Data Diagnostic
2. Delete Session
3. Session is Used
4. File Storage Failure
5. Remove Evaluation Users
6. List Sessions

0. Exit

Enter your choice : 6

2.5 List Sessions

This operation lists all the evaluation sessions recorded in the TRDB. The data displayed is : the session name, the session state (CREAted, ARCHived, ONLIne, To Be ARchived, To Be EXported, To Be IMported, To Be Completely Deleted, To Be Partially Deleted), the session owner and the session creator.

NAME	*STAT*	*OWNER*	*CREATOR*
EVAL_BEN_001	*ONLI*	*rt_user3*	*rt_user1*
EVAL_BEN_003	*CREA*	*rt_user3*	*rt_user3*

Figure 9–41 : *Evaluation Session List.*

9.2.1.3 DBS Error Number (DBS_ERR_xxx)

Select the ’DBS Error Number’ operation by entering ’3’ in the Main Menu. This operation provides some information on the high level error messages issued by central DBS. The high level messages sent by Central DBS contain an identifier. By entering this identifier, the user can obtain further explanations on the consequences and the possible recovery actions following the problem detected. Two examples are given at the

Copyright per DIN 34

Figure 9–42 It shows also the DBS error message sent to the CGSI for the first example.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 311
<DBS_ERR_311>
Storage of an Event file into the TRDB failed.
Central DBS has not been able to create an Oracle reference for this Event file or it failed to insert the file content into the Oracle event table.
The temporary file should be deleted and the Oracle table references updated, using the DBS Recovery Scripts <File Storage Failure> Menu.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_E

Error message sent by DBS to CGSI:
DBS_ERR_311: Local Event file not stored into the TRDB [Recovery 1.4]
<file name>, <session name>.

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 213
<DBS_ERR_213>
The Import of a Test Execution Session from FA medium failed.
The Import of the Oracle Table containing Eng. Value file references for the given Test Execution Session failed because the file where this table has been exported has a wrong format.
Data and references are left on the TRDB disk.
The imported session (NEW_NAME) should be deleted by the Recovery Scripts <Session is Used> Menu.

Copyright per DIN 34

Figure 9–42 : *DBS Error Number Explanations*

If the DBS error number selected is out of range, an error message is printed as shown on the Figure 9–43

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 145
Invalid DBS Error Number.

– DBS RECOVERY SCRIPTS MAIN MENU –

1. Execution Session
2. Evaluation Session
3. DBS Error Number (DBS_ERR_xxx)

0. Exit
Enter your choice : 3

3. DBS Error Number (DBS_ERR_xxx)

Enter the DBS Error Number that you want to be explained (3 digits): 938
Invalid DBS Error Number.

Figure 9–43 : *DBS Error Number Explanations*

Copyright per DIN 34

10 CGS ADMINISTRATION

CGS Administration consists mainly of the following tasks:

- System Administration:
  Adding and removing workstations and test nodes to/from a CGS system
- User Administration:
  Adding and removing users from CGS or modifying the user privileges
- Defining the system topology and adapt the CGS configuration parameters
- Maintaining the Storage Resources
- Exporting and importing MDB data and TRDB data
- Startup/ Shutdown of the system and monitoring the system behaviour

10.1 CGS System Administration

10.1.1 CGS Component Version Status Info

Installed CGS component versions status can be checked by the CGS Task Selector —> Software Versions.

10.1.2 CGS File System Structure

The CGS file system structure is shown in the following tables. The definition of the CGS configurable scope concept can be found in Appendix K.

```
/var/opt/cgs
        |
        |— login.csh  (owner=root, –rw–r—r—)      — contains CGS_HOME= , CGS_ARCH=
        ‘— login.sh                               — contains CGS_HOME= , CGS_ARCH=


      $CGS_HOME
        |
        |— bin                          — links to startup scripts (to be established)
        |
        |— doc                          — to be established (manuals, ...)
        |
        |— etc                          — global data (CGS configuration files, image files, ...)
        |  |
        |  |— mda                       — generated global mda data
        |  |  |— ccu
        |  |  |— i_code
        |  |  |— mdb_installations
        |  |  ‘— test
```

Copyright per DIN 34

```
|  |
|  `— project                    — project specific login.sh AND login.csh
|                                — (to be generated by project)
|
|                                — project specific pre_generate_scoe and
|                                — post_generate_scoe (called by cgs generate scoe)
|                               — (to be implement)
|
|— gsaf                          — CGS software
|  |— cgsi
|  |  |
|  |  |— lib                     — CGS libraries for:
|  |  |  |
|  |  |  |— api                  — ADA – API
|  |  |  |  |— C
|  |  |  |  `— ada
|  |  |  |     |— linuxi_gnat
|  |  |  |     `— sun5_gnat
|  |  |  |
|  |  |  |— corba                — systeminterface.idl
|  |  |  |
|  |  |  `— ucl                  — UCL libraries
|  |  |
|  |  `— user_env                — global user definitions
|  |
|  |— cis
|  |— cls
|  |
|  |— data                       — to be clarify ?
|  |
|  |— dbs
|  |— gwdu
|  |— hci
|  |
|  |— manual                     — to be (re)moved
|  |
|  |— sas
|  |  `— bin                     — SAS executable/startup scripts  (to be moved?)
|  |
|  |— tes
|  |— tev
|  |— tscv
|  `— tss
|
|— local                         — host (Box) depend data
|  |
|  |— config                     — CGS configuration files for host,
|  |                            — overwrite global CGS configuration files
```

Copyright per DIN 34

```
|  |                          — SYSTEM_TOPOLOGY_TABLE
|  |
|  |— data                    — TES error data (to be moved ?)
|  |  |— log                  — host log data (install.log, message handler logs)
|  |  '— trdb                 — TRDB
|  |
|  |— etc
|  '— tmp                     — temporary data (to be removed after CGS shutdown ?)
|     |— log
|     |— mda
|     |  |— ccu_default
|     |  '— log
|     |— pcs
|     |— tes
|     '— tss
|
|— patches                    — CGS patch directory
'— tools                      — commercial tools delivered by CGS
   |— adobe
   |— ghostscript
   |— gsview
   |— java_utilities
   |— jre–1_2
   |— jre–1_3
   |— jre–1_4
   '— tcltk
```

Table 10–1 : *CGS System File Structure*

Copyright per DIN 34

```
$HOME/.cgs                          — user depend CGS files
        |
        |
        |                               — otc.properties (old USER_PROFILES)
        |
        |
        |— bde                          — directory for batch data entry (export/import)
        |  |— input
        |  ‘— reports
        |
        |— cgsi                         — to be clarify ?
        |
        |— config                        — to be established
        |                               — CGS configuration files for user,
        |                               — overwrite global CGS configuration files
        |
        |— screen_setup_pool                  — HCI screen setups for user
        |
        ‘— tev                          — TEV user results
           |— DEFINITIONS
           |  |— DATA_LIST
           |  |— DATA_SET
           |  |— EVENT_LIST
           |  |— GRAPHS
           |  |— RAW_DATA_DUMP
           |  ‘— STATISTICS
           |— EXTERNAL
           |  |— DATA_LIST
           |  ‘— DATA_SET
           |— RESULTS
           |  |— DATA_LIST
           |  |— DATA_SET
           |  |— EVENT_LIST
           |  |— GRAPHS
           |  |— MERGED_DATA_SET_EVENT
           |  |— RAW_DATA_DUMP
           |  ‘— STATISTICS
           |— SAS
           ‘— SESSIONS
```

Table 10–2 : *CGS User File System Structure*

10.1.3 CGS Configuration Support Tool

A GUI–supported CGS configuration tool is located at:

$CGS_HOME/gsaf/config/bin/configuration.tcl

This tool supports also the tasks described below.

Copyright per DIN 34

10.1.4 CGS Hardware Installation/Deinstallation And Booting

To turn on/boot the CGS hardware it is recommended that the database server host is turned on/booted first. When the boot sequence has finished (login prompt/window visible), the workstations, test nodes and simulation nodes can be turned on/booted in arbitrary order.

- *Adding and deinstalling workstation clients, test nodes and simulation nodes are described in the CGS Installation manual (see 2.4).*

10.2 CGS User Administration

10.2.1 Add CGS User

- *Refer to the CGS Installation Manual (see reference in 2.4).*

The dedicated skrip requires cgsadmin rights and can be found at

$CGS_HOME/cgs/util/common/install_user

10.2.2 Deinstall CGS User

- *Refer to the CGS Installation Manual (see reference in 2.4).*

10.2.3 CGS User Authorization

The CGS user authorization concept consists of system and user privileges that needs to be assigned to each CGS user. This CGS generic authorization concept needs to be mapped to application projects as outlined below.

All CGS user authorization information are located in a central privileges file. Write access is only granted to the cgsadmin user. CGS provides a tool named *priv* tool to manage the system privileges (as described in 10.2.3.3).

10.2.3.1 CGS System Privileges

CGS system privileges (former CGS user roles) needs to be assigned to all CGS user. Each CGS user has one or more system privileges.

System privileges define what the user may do with the CGS system or with specific subsystems.

Currently there are three groups of CGS system privileges:

- CGS privileges
- CIS privileges
- MDB privileges

The CGS privileges define the user's authorization with respect to the usage of the CGS system as defined in Table 10–3.

CGS system privilege	Description and Responsibilities
CGS:SETUP	Test system setup

Copyright per DIN 34

CGS:OPERATE	Online control through HLCL commands
CGS:EVALUATE	Evaluation (observe results)
CGS:STARTUP	System configuration, startup of the CGS system
CGS:SHUTDOWN	System configuration, shutdown of the CGS system

Table 10–3 : *CGS privileges*

MDB privileges define the user's authorization with respect to the usage of the mission database MDB as defined in Table 10–4.

MDB privilege	**Description and Responsibilities**
MDB:CONFIGURE	Creation and maintenence of mission configuration data
MDB:USE	Change the content of data entities

Table 10–4 : *MDB privileges*

CIS privileges define the user's authorization with respect to the usage of the external CGS interface server (CIS) as defined in table below. They have been introduced to save the SendTc routine of the Commanding interface (CGS CORBA IDL). Only users with the correct privileges (shown below) can send telecommands via CIS. Verify with $CGS_HOME/gsaf/cgsi/bin/common/priv privileges.

CIS privilege	**Description and Responsibilities**
CIS:SENDTC	user is able to send telecommands via CIS
CIS:SENDSWOP	user is able to send SWOPs via CIS
CIS:SENDFLAP	user is able to send FLAPs via CIS
CIS:USE_HLCL	user is able to execute HLCL commands via CIS

Table 10–5 : *CIS privileges*

The mapping of old user roles to new system privileges is shown below:

system privilege	**old user role**
CGS:EVALUATE, CGS:OPERATE, CGS:SETUP	CGS:test_conductor
CGS:EVALUATE, CGS:OPERATE	CGS:test_operator
CGS:EVALUATE	CGS:test_evaluator
MDB:CONFIGURE	MDB:configuration_manager
MDB:USE	MDB:normal_user

Table 10–6 : *mapping system privileges – old user roles*

Copyright per DIN 34

10.2.3.2 CGS User Privileges

Privileges can be defined by an application project.

Privileges define the user's access rights to single database end items as defined and controlled by a mission database, such as commands, measurements, variables, AP's, libraries, etc.

Privileges are assigned to user roles by a dedicated assignment tool as described in 10.2.3.3.

- *Within end item libraries, privilegers can be assigned to guarded procedures and functions. Parameters in procedures and functions can also be guarded. This allows checking of priviledges during run time of AP's. Write access to SW variables in AP's is guarded by default. Imported libraries within AP's are always guarded.*

Privileges are compiled during the compliation of an AP by considering assigned and inherited privileges. Privileges can be inherited by referencing other entities with privileges assigned.

Privileges are validated during the invocation of AP's in the HLCL interpreter, where violations against user assigned privileges are reported.

Example:

Privileges of telecommands are validated during an "Issue TC Command" by the HLCL interpreter.

10.2.3.2.1 Syntax of Privilege Descriptions

Privilege identifier are case insensitive with the lexical from:

Letter { ["_"] (Letter | Digit) }

10.2.3.3 The PRIV Tool for Privilege Assignments

Assigning and de–assigning privileges to user roles is supported by a dedicated Unix PRIV tool. The tool has a command and a graphical user interface. It can be configured by a (XML based) user authorization file.

The authorisation for usage of this tool requires the cgs–admin Unix system access rights.

The command interface to the priv tool follows the syntax:

priv subcommand [parametres] [–options]

Command help can be printed by the –help option. Subcommands and options are summarized in Table 10–7.

Subcommand/Parameter	**Comment**
define <privileges>	define new privileges
undefine	remove defined privileges
add <user> [–role <value>]	add a user; requires user name and user roles
remove <user>	remove a single user; '*' remove all users
roles [<user> –role <value> ...]	show or set user roles (all or single user)
privileges [<user>]	list current privileges of all or single users
users [–role <value> ...][–privilege <value> ...] [–long]	list current users for single roles or privileges long output format
grant <user> <privileges>	grant or reset privileges for single or all users

Copyright per DIN 34

withdaw <user> [<privileges> ...]	withdraw privileges for single or all users
export <file> [–user <value> ...][–role <value>...]	export user definitions to user defined file
import <file> [–reset]	import privileges fro file; reset users and privileges first
check <users> [<privileges> ...]	check for user and his privileges; return status
mdb <item> [–environment <value> [–list] [–add_assigned <value> ...] [–add_library <value> ...] [–set_assigned <value> ...] [–set_library <value> ...]	

Table 10–7 : *Subcommands for the PRIV Tool*

Two different GUI's to the priv tool can be invoked by the commands:

priv

priv mdb pathname

Figure 10–3 and 10–4 show example applications on allocation of privileges to an end item or end item library.

Copyright per DIN 34

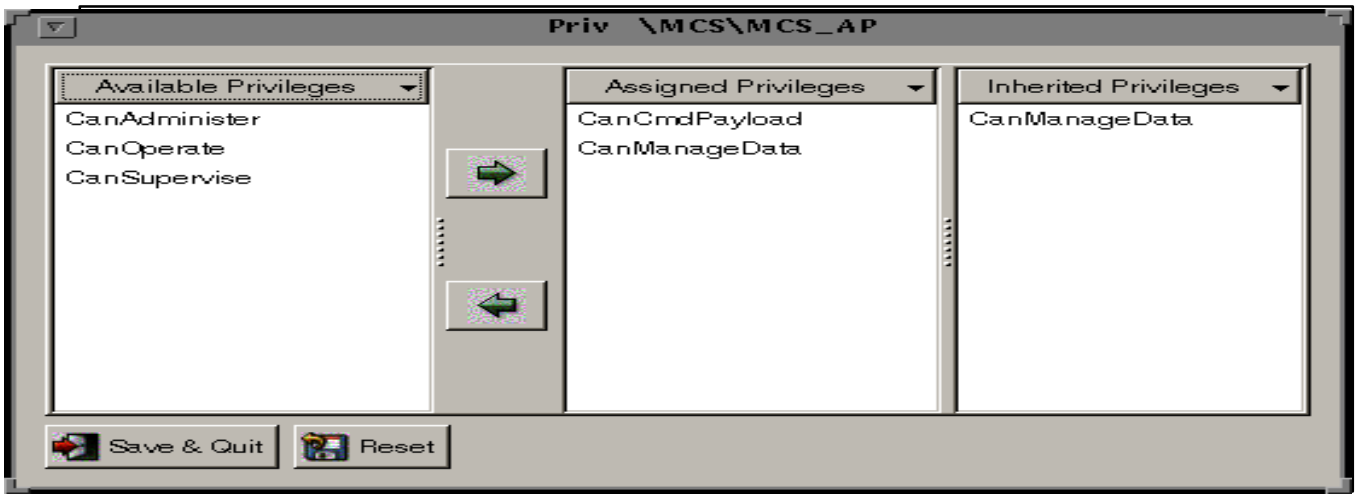


Figure 10–1 : *Example GUI for End Item Pirvilege Allocation*

Copyright per DIN 34

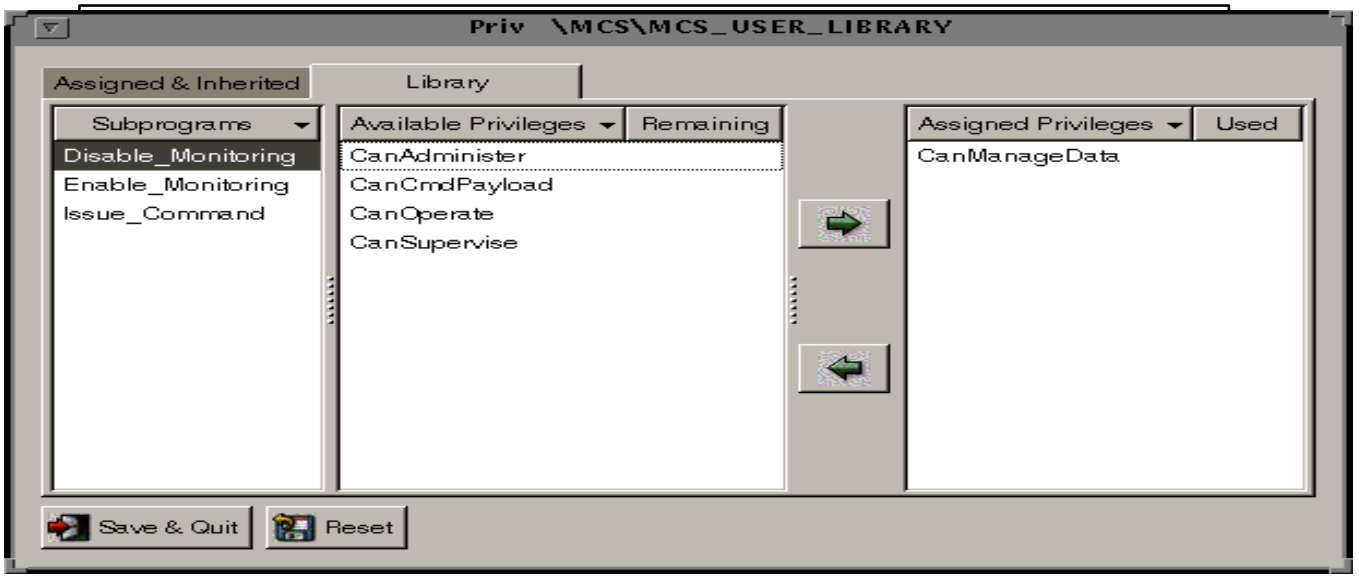


Figure 10–2 : *Example GUI on Library Privilege Allocation*

- *Details are tbd*

10.3 Configuration Setup

10.3.1 Modify System Topology Table

The sytem topolpgy table defines a mapping between logical host names to physical host names for a CGS (test) site. The system topology file is located in $CGS_HOME/local/config/.

Whenever a test node or workstation has been added/removed, the System Topology Table needs to be changed.

One way of modifying the System Topology Table is defined in the CGS Installation Manual.

Another way of changing the table is provided with TSCV by opening the TSCV tool and select the 'Properties –> System Topology' menu option. Refer to ch. 7.2.3.6 of this Manual.

10.3.2 Maintain CGS Configuration Parameters

CGS defines configuration parameters in several configuration files. The scope of these parameter definitions may be global, local or user defined. The configuration parameter can be changed by modifying the configuration files with an text editor.

- *For an overview of the files that may be modified see appendix K.*

Copyright per DIN 34

10.4 Oracle Startup/Shutdown

The Oracle Server Startup / Shutdown procedure is described in the Oracle manuals, and the CGS Administrator should refer to this.

Usually there is an startup / shutdown under /etc/init.d/oracle, which is foreseen to be called when the server is booted resp. when it is shutdown, but it may be used also for manual start-up/shutdown of the Oracle processes.

To call the shutdown of Oracle, enter as root user in a command window:

/etc/init.d/oracle stop

To startup, enter

/etc/init.d/oracle start”

10.5 MDB Administration

The following is a short overview on administration tasks for the MDB. The tasks may be allocated to different users, thus implementing distribution of CGS administration to specific experts.

To change the privileges of the user’s a deinstallation and re–installation with the new privileges is recommended via $MDA_HOME/config/oracle_env/deinstall_user resp. $MDA_HOME/config/oracle_env/install_user (refer to MDA Administration Manual)

10.6 Maintain Storage Resources

10.6.1 Resource Considerations

When performing a test system setup and/or test session creation, the user has to especially consider the amount of disc space and database table space consumed during the test. This is of special importance because CGS has no chance to continue properly in case the discs are full or the database has no space left. Data has to be throughn away and will then be lost for the user.

- general session information (master archive) for execution and evaluation sessions,
- events generated at test execution,
- engineering value logbooks generated at test execution,
- archive files (raw data) generated at test execution,
- and result files being generated during test evaluation (e.g. event lists, data sets etc.).

Not all data will be physically stored within the ORACLE RDBMS as tables, but within the UNIX filesystem. General session information and events are directly stored as rows within ORACLE tables. All other type of data will be stored as files within the UNIX Filesystem, where the ORACLE RDBMS controls references to these files through its UNIX pathname.

Copyright per DIN 34

10.6.2 Hard Disc (Magnetic Disc)

As described before, parts of TRDB data for execution and evaluations sessions are stored in the UNIX file-system under the following location:

$CGS_HOME/local/data/trdb.

Beneath this directory, a structure is created for:

- a directory for storing EXECUTION session files,
- a directory for storing EVALUATION session files,
- a WORK directory for temporary scratch pad use.

Space under these directories is consumed whenever:

- a test execution session is initialised or a test evaluation session is created,
- raw data files are stored in TRDB, located in $VICOS_CEN_DBS_HOME
- engineering result files are stored in TRDB,
- evaluation result files, configuration files, event files and application supplied files are stored in TRDB,
- a session can be retrieved from the optical disc archive media to the TRDB

Space under EXECUTION and EVALUATION session directories is released whenever

- a test execution session or a test evaluation session is deleted,
- an evaluation result file is deleted,
- a session is archived.

It has to be made sure that there is enough space for these files. In case of space problems, repartioning might be necessary.

Repartioning of the UNIX filesystem requires specific system administration / root priviledges and is not further explained here. Please refer to the operating system documentation.

Other disk space consuming files are

- message log files in $CGS_HOME/local/data/log
- large models stored in the file system (not used in V6.x)
- MDB reports
- Load_Scoe files and associated listings under $CGS_HOME/etc/mda
- MDB Export files and BDE files

A system housekeeping variable MD_FREE_SPACE (ID=1011) is available on DBS node to obtain the amount of free disc space for TRDB. This variable refers to the number of available kilo bytes in the file system holding the directory $VICOS_CEN_DBS_HOME/...

10.6.2.1 Monitoring of Disc Space

A system housekeeping variable FREE_DISC (ID=90) is available on each test node to obtain the amount of free disc space locally available on the test node. This variable refers to the number of available kilo bytes in the file system holding the directory $TN_HOME/...

Copyright per DIN 34

In case the system is working normally, archive files will be transfered automatically from $TN_HOME to the file system of the central database server. Thus HK variable 90 should be fairly constant running CGS. In case the archive files cannot be stored on the central TRDB disc any longer, the value should decrease dramatically by 30 MByte/30 minutes. This is an indication for disc space overflow and normally the test should be stopped immediately since CGS cannot guarantee that no data are lost.

Housekeeping values can easily be monitored by defining a SW variable in MDB for each test node which maps to this HK value and which has appropriate limits and associated actions defined.

10.6.2.2 Delete/Export Test Sessions

CGS Administration should monitor the resources used by the TRDB. A regular check should be made, if test sessions can be deleted or exported.

Deletion of test sessions is provided via the TSCV menu option ”Test Session –> Maintain” and the ”Maintain Test Session” Subwindow. Deletion within the default test session should be taken into account as well. The ”Maintain Test Session” Subwindow provides the respective selection criteria and the ”Delete in Default Session” option.

If a test session is not in state ’open’ or ’closed’ but in an inconsistent or error state, the user via TSCV might only be able to list the session (by deselecting both ”open” and ”closed” for Session Status in the ”Maintain Test Session” window of TSCV), but not to delete them. In this situation, the DBS recovery scripts need to be called.

To export a test session to an optical disk, the TEV Export/Import Tool can be used.

If a failure occurs during export, again the DBS Recovery Scripts must be called.

10.6.2.3 CleanUp of Disk Memory

When disk space is below a certain amount of free blocks, or on a regular basis, the CGS administrator needs to cleanup the disks.

Depending of the amount of archive files and engineering value log files stored in test sessions, deleting or exporting of test sessions will also free the disks where $GSAF_HOME is located.

To support further cleanup of the disks, , the script

`$CGSI_HOME/bin/common/cleanup`

is provided. It allows to remove or compress CGS log files and temporary files as well as archive files. It provides for selection of the type of files to be removed/compressed.

Before calling the script, the CGS Administrator must verify, that no CGS Processes are running in the system. A secure approach is to shutdown CGS before using the script.

The following is a sample output of the cleanup script:

```
-------------------------------------------------------------------
-- Cleaning up CGS related files
```

Copyright per DIN 34

```
-- NOTE: CGS should not be running, shutdown if running!
-- Type s or RETURN to skip, r for removal or c for compression !
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-- /opt/cgs/local
                                  data/Event_Distributor_Daemon*.log [s] r
                                                 data/log/*.log [s] r
                                               data/log/*.log.Z [s] r
                                                 data/log/*.ack [s] r
                                               data/log/*.ack.Z [s] r
                                                    data/nwsw/* [s] r
                                              data/cmas/logger/* [s] r
                                                 data/trdb/*.log [s] r
                             data/trdb/WORK/DEFAULT_TEST_SESSION/event/* [s]
                          data/trdb/WORK/DEFAULT_TEST_SESSION/eng_value/* [s]
                                           data/trdb/WORK/*.EVT.* [s]
                                           data/trdb/WORK/*.EVL.* [s]
                                           data/vicos_tes.output* [s] r
-- /opt/cgs/local/tmp
                                                      ./tscv.lock [s]
                                                     ./menu.log [s] r
                                                       ./*.read [s] r
                                                      ./*.write [s] r
                                                     ./*.script [s] r
                                                      ./*.trace [s] r
                                                        ./*.tmp [s] r
                                                    ./dbs_*.log [s] r
                                                          log/* [s] r
                                                    mda/log/* [s] r
                                                        tes/* [s] r
                                               ./vicos_tes.output* [s] r
                                                        pcs/*.log [s] r
                                                       ./tev_* [s] r
-------------------------------------------------------------------------
-- /opt/cgs/etc
                                             mda/test/load_scoe.* [s] r
                                                  mda/test/*.log [s] r
                                                  mda/test/*.lst [s] r
                                                 mda/test/*.tree [s] r
                                                      mda/log/* [s] r
-------------------------------------------------------------------------
What testnode directories (also server and workstations possible)
Enter all nodes seperated by a blank in one line
Node(s) [leave blank for none]:
-- /tn
-------------------------------------------------------------------------
-- /tmp
```

Copyright per DIN 34

```
                                                    ./DDED_*.msg [s] r
                                                     ./PNS_*.msg [s] r
```

When started with statistic option

```
$CGS_HOME/bin/common/cleanup –statistics
```

information about number of removed and compressed files is provided like

```
.
.
-- /tmp
                                                        ”adatmp*” [s]
                                                   ”*.sw_versions” [s]
-------------------------------------------------------------------------
            Removed: 242
         Compressed: 0
               Gain: 2358 blocks (a 512 Bytes)
-------------------------------------------------------------------------
```

A clean up of temporary disk space is performed during shutdown of CGS.

10.6.3 Configuration of Sizes and Estimation of TRDB Data

10.6.3.1 Size Configuration for Event Logging

Events logged by an application are first of all written into a pool of buffers. These buffers are cyclically read by an background task within the application, that copies them into a temporary file. After that, this file containing events is sent to central DBS (before loading into the ORACLE RDBMS). The temporary event file is transferred:

- when it reaches a given size (defined by DBS.RPI.MAX_EVT_NUMBER_IN_LOCAL_FILE),
- periodically (defined by DBS.RPI.ONL_EVAL_EVT_PERIOD),
- when the session in which the application takes part is closed.

Both parameters have been pre–configured in $CGS_HOME/etc/cgs_configuration.xml:

– MAX_EVT_NUMBER_IN_LOCAL_FILE = 1000 <hidden>

Max Number of Events in a local file (before it is sent to central).

Recommended value: 1000

– ONL_EVAL_EVT_PERIOD = 10 <hidden>

The period (in seconds) between local EVT file storages on the Central DBS server. Then they will be available for online evaluation.

Recommended value: 10

Single events are stored as rows within ORACLE tables. The size of a single event can vary, depending on the length of the short and long text field. Both is application dependant. According to own experience, the average size of a single event is 120 Bytes.

Copyright per DIN 34

The minimum size of a temprorary file is then the size of one event (ie. appr. 120 Bytes). Assuming that MAX_EVT_NUMBER_IN_LOCAL_FILE has been reached, the maximum size of the temporary event file before loading into ORACLE is then 450 * 120 Bytes = 54 KBytes.

10.6.3.2 Size Configuration for Engineering Value Logbooks

Engineering values logged by an application are first of all written into a pool of buffers. These buffers are cyclically read by an background task within the application and their content is written into a file. After that, the file containing engineering values is sent to central DBS:

- when it reaches a given size (defined by DBS.RPI.MAX_EVL_NUMBER_IN_LOCAL_FILE),
- periodically (defined by DBS.RPI.ONL_EVAL_EVL_PERIOD),
- when the session in which the application takes part is closed.

Both parameters have been pre–configured in $CGS_HOME/etc/cgs_configuration.xml:

– MAX_EVL_NUMBER_IN_LOCAL_FILE = 2000 <hidden>

Maximum Number of Engineering Values in EVL Local Files.

Recommended value: 2000

– ONL_EVAL_EVL_PERIOD = 10 <hidden>

The period (in seconds) between local EVL file storages on the Central DBS server. Then they will be available for online evaluation.

Recommended value: 10

According to own experience, the size of an engineering value can vary between 38 and 547 Bytes.

The minimum size of a file is then the size of one engineering value (i.e. 38 bytes), and the maximum size, as defined by the default configuration value, is 450 * 547 Bytes = 246.15 Kbytes.

10.6.3.3 Configuration and Size Confuiguration of Archive Files

An archiving device is used for archive, export, retrieval and import CGS operations. The CGS archiving device requires a mounted Unix file system.

Instructions to create a Unix file system, to mount file systems and devices and to initialise the related disc devices are given in the CGS Installation Manual.

Theoretically there is no limitation for CGS archiving operations. A record of all data transferred to the archiving device is maintained in the TRDB master archive tables.Whenever an optical disc is full, the FA SAS will automatically ask for a new disc to be inserted.

The size of the archive files depends on the amount of data being archived per second and the time intervall an archive file is open until the next one is created.

Under nominal data rate conditions (80 kbits telemetry, 20 packets per second plus 3 kbits telecommand, 20 packets per second) and assuming the default time period for archiving (30 minutes) the size should be below 30 MByte.

The file system in which the $CGS_HOME/local/data/trdb directory resides must have sufficient free space to hold the archive files from all test nodes included in a session over the time period planned for the test session and obeying the data rates.

Copyright per DIN 34

10.6.3.4 Result File Size and Handling

Evaluation result files will be saved under the users' home directory, in particular under ~/wd/tev/..... Therefore first–of–all the disc quota limitations applied to a user will put constraints. Through the test evaluation functionality of CGS it is possible to directly store a result file into the TRDB. By doing this, the result file will be moved from the users' directory to the location of the TRDB filesystem.

The size of an evaluation result file is non–predictable. It mainly depends on the selection criteria which have been applied (e.g. time frame, number of measurements, ADUs, GDUs selected etc.). Therefore no estimates can be given here.

10.7 Monitor System Behaviour

10.7.1 Monitor Process Status

To monitor the UNIX process status on a CGS node, the normal UNIX commands can be used.

Furthermore, CGS provides some predefined tools to get the process status list.

Login to a CGS node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via:

```
$CGSI_HOME/bin/common/ts&
```

The menu option "CGS Process Status" in the menu button group System Status gives you a list of all CGS related active processes on the related node.

Sample Output List (for a DB Server node):

```
   98 /usr/lib/netsvc/yp/ypbind
   95 /usr/lib/netsvc/yp/ypserv
  106 /usr/lib/netsvc/yp/ypxfrd
  111 /usr/lib/netsvc/yp/rpc.yppasswdd
15219 vicos_tss_tsp_sun5
15307 dbs_central_eval
15258 dbs_central_arch
15335 dbs_central_exec
 1188 rpc.ttdbserverd
27930 ora_lgwr_oracle
 8690 oracleoracle
 6154 oracleoracle
27926 ora_pmon_oracle
  427 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/tnslsnr
 7026 oracleoracle
15433 oracleoracle
15419 oracleoracle
27932 ora_smon_oracle
15302 oracleoracle
28224 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/maid.SunOS5.4.release
```

Copyright per DIN 34

```
28223 /gsaf_home/oracle_home/app/oracle/product/7.3.2/bin/oraweb
 6144 oracleoracle
29215 oracleoracle
 8683 oracleoracle
27928 ora_dbwr_oracle
```

The number preceding the process name denotes the process identifier PID.

Further information can be obtained for each process via the command:

”ps –f –p <PID>”

to be invoked in a UNIX command window.

Refer to Solaris Manuals for the output format or type ”man ps”.

10.7.2 Monitor Memory Status

Login to a Unix node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via:

$CGSI_HOME/bin/common/ts&

The menu option ”Memory Status” in the menu button group System Status gives you a window with information on the memory used on that node:

Memory usage for host linpc3 at time Wed Mar 10 15:09:27 CET 2004

	total	used	free	shared	buffers	cached
Mem:	**1028920**	**954600**	**74320**	**132120**	**51224**	**418716**
–/+ buffers/cache:		**484660**	**544260**			
Swap:	**2040244**	**459548**	**1580696**			

Copyright per DIN 34

10.7.3 Monitor Time Synchronisation Status

Login to a Unix node as cgsadmin user, set the DISPLAY variable to your workstation and call the CGS Task Selector via:

```
$CGSI_HOME/bin/common/ts&
```

The menu option ”Time Protocol” in the menu button group System Status gives you a window with access to the ntpq tool.

This ntpq tool allows to get information on time synchronisation between the CGS nodes.

The ntpq program is activated which allows for the following commands:

```
addvars         associations   authenticate     cl           clearvars
clocklist       clockvar       cooked           cv           debug
delay           help           host             hostnames    keyid
lassociations   lopeers        lpassociations   lpeers       mreadlist
mreadvar        mrl            mrv              ntpversion   opeers
passociations    passwd         peers            poll         pstatus
quit            raw            readlist         readvar      rl
rmvars          rv             showvars         timeout      version
writelist       writevar
```

To obtain a list of in–spec peers of the server, along with a summary of each peer’s state invoke the ntpq command:

Peers

Summary information includes the address of the remote peer, the reference ID (0.0.0.0 if the refID is unknown), the stratum of the remote peer, the polling interval, in seconds, the reachability register, in octal, and the current estimated delay, offset and dispersion of the peer, all in seconds.
In addition, the character in the left margin indicates the fate of this peer in the clock selection algorithm.

Characters only appear beside peers which were included in the final stage of the clock selection algorithm. A ”.” indicates that this peer was cast off in the falseticker detection, while a ”+” indicates that the peer made it through.

A ”*” denotes the peer the server is currently synchronizing with.

Note that since the peers command depends on the ability to parse the values in the responses it gets it may fail to work from time to time with servers which poorly control the data formats.

Copyright per DIN 34

A ACRONYMS

A

ADU	*Acquisition Data Unit*
AIV	*Assembly, Integration and Verification*
AP	*Automated Procedure*
API	*Application Program Interface*
APM	*Attached Pressurized Module*

B

BDE	*Batch Data Entry*

C

CCU	*Configuration Control Unit*
CCSDS	*Consultive Comitee of Space Data Systems*
CDU	*Configuration Data Unit*
CGS	*Columbus Ground Software (or System)*
CGSI	*Columbus Ground System Infrastructure*
CI	*Configuration Item*
CIS	*CGS Interface Server*
CLS	*Columbus Language System*
CM	*Configuration Management*
CORBA	*Common Object Request Brocker Architecture*
CSS	*Core Simulation Software*

D

DBB	*Data Base Browser*
DBMS	*Data Base Management System*
DBS	*Data Base Services*
DDED	*Detailed Data Editor*

E

EGSE	*Electrical Ground Support Equipment*

F

FES	*Front End Software*

Copyright per DIN 34

FLAP	*Flight AP (Automated Procedure)*
FWDU	*Flight Window Definition Utility (obsolete)*

G

GDU	*Generation Data Unit*
GUI	*Graphical User Interface*
GTAP	*Ground Test AP (Automated Procedure)*
GUI	*Graphical User Interface*
GWDU	*Ground Window Definition Utility*

H

HCI	*Human Computer Interface*
HK	*House Keeping*
HLCL	*High Level Command Language*
HW	*Hardware*

I

ICD	*Interface Control Document*
ICP	*Immediate Command Processor*
ID	*Identifier*
I_MDB	*Integrated Mission Data Base (GUI)*
I/O	*Input / Output*

J

K

L

LACIS	*Logiciel Adapte a la Comuunication des Informations Systemes. (Project specific format)*
LAN	*Local Area Network*

M

Copyright per DIN 34

MBF	*Mission Build Facility*
MDA	*Mission Database Application*
MDB	*Mission Data Base*
MDE	*Model Development Environment (obsolete)*
MOCS	*Model Observation and Control System (obsolete)*
MPS	*Mission Preparation Software*

N

NWSW	*Network Software*

O

OB	*On-Board*
OS	*Operating System*

P

PL	*Payload*
PUS	*Packet Utilisation Standard*

Q

R

S

SAS	*Special Application Software*
SDE	*Software Development Environment*
SID	*Short Identifier*
SITE	*Software Integration and Test Environment*
SIVQ	*Software Integration and Verification*
SMT	*Simulated Mission Time*
SWEU	*Software Exchangeable Unit*
SWRU	*Software Replaceable Unit*

T

TBC	*To be confirmed*

Copyright per DIN 34

TBD	*To be defined*
TBS	*To be supplied*
TC	*Tele Command*
TC	*Test Configuration*
TES	*Test Execution Software*
TEV	*Test Evaluation Software*
TL–UI	*Top Level User Interface*
TM	*Telemetry*
TPS	*Technical Publishing Software (Interleaf)*
TSCV	*Test Setup, Configuration and Verification software*
TSS	*Time Synchronization Software*

U

UCL	*User Control Language*
USS	*Unified Synoptic System*

V

VICOS	*Verification, Integration and Checkout Software*

W

WDU	*Window Definition Utility*

X

Y

Z

Copyright per DIN 34

B DEFINITIONS

A

Abstract Data Type(ADT) A set of procedures and functions encapsulating a complex data type. Through the procedural access instances of the data type may be created and destroyed and individual components may be accessed for reading and writing.

Access rights Define what access various users or applications have to objects or entities.

Acquisition Data Unit (ADU) An ADU is a data unit that is received by VICOS, which contains data (enditem values) to be calibrated and monitored. In general ADUs contain a set of enditem values in raw format. ADUs may be structured (i.e. a list of enditem values) or unstructured (i.e. a byte array where the values are packed, or a TM packet).

Action In CGS, the term action is used to describe activation of APs, sending of commands(GDUs), generation of messages or enabling of processing as a result of monitoirng exceptions or conditions.

Application Program or set of programs performing some specialized user–oriented function (as opposed to general–purpose programs like a DBMS, or an Operating system)

Application Home The directory location assigned to a particular application of CGS, also called Home Directory.

Application Independence Application independence is the software characteristic that ensures that the software is not dependent on any database system, microcode, computer architecture or algorithms.

Archive Refers to the process of relegating obsolete data to external backing storage. The reverse operation (copying archived data back to active storage) is known as restore.
In CGS, either MDB contents may be archived, or complete test sessions.
In VICOS, archiving in addition means storage of data in raw format. All data received or generated by a testnode is archived in OS files (archive files). The files are managed after closure as part of the Test Result DB and made accessible by Test Evaluation S/W.
In VICOS, data storage to disks is called ”Final Archive”. Refer to ”Final Archive”

Atomic Function Block (AFB) To specify a model in CSS, a simple block having inputs and ouputs defined is used as one of the basic definitions. Such an item is called Atomic Function Block and can be seen as mapped to one procedure of the implementation language AIL.
Two kinds of AFBs are defined: Asynchronous AFBs, which are activated by events, and synchrounous AFBs, which are activated periodically in each simulated time frame.

Copyright per DIN 34

Atomic Implementation Language (AIL)
Language used for the implementation of atomic function blocks in a model definition.

Automated Procedure
A program written in the User Control Language (UCL)**.**
It is a ”compiled program in UCL” which correspond to programs in the usual sense, i.e. which may be not imported by other modules but which may be executed directly.

B

Baseline
A set of explicitly defined document issue/revisions, CI constituent versions and lower level CI instantiation baselines, which is used for a CI instantiation.

Base Function
The basic function underlying a function block, which is used in associated with the function mask to provide the function block, c.f. Base Function and Function Mask. E.g. An analogy is the invocation of a C compiler with a set of options; the *function block* is the C compiler invoked by ”cc”, the *function mask* is the specified set of options ”–O”, ”–target”, etc. which specify parameters particular to the invocation (”optimisation”, ”cross–compilation”, etc.).

Batch Mode
A program operationg in batch mode executes all its actions wothout intervention by a human operator.
In CGS, the programs TEV and TSCV can be operated in batch mode, in addition to the normal interactive mode.
Batch Operations are also available for MDB data entry/export via the Batch Data Entry (BDE) Tool.

C

Calibration
Calibration means calculation of values for enditems according to predefined calibration curves, from raw format (i.e. format as received resp. acquired via devices) into calibrated format. The result of a calibrated value is the Engineering Value, which may imply an Engineering Unit to be applied.
CGS extends this definition by referring to calibration also in cases, where a string is extracted from raw data and where state codes are generated for raw discrete values.
Decalibration is the reverse process (–> Decalibration)

CDU domain
is a set of MDB item types. CDUs have a specific subset of all MDB item types defined, to allow for working in a specific context of data definitions.

CGS Administrator
The user administrating the CGS Installation and the Setup of the system. Its a specific user, intalled as CGS User, with specifc writes. Owns all of the CGS Software executables.
Is allowed to startup and shutdown the system and to define new users.

Copyright per DIN 34

CGS Server Node in the network that provides file server functions and executes the Oracle Services. In a standard setup, this node also exutes the DB Server node functions, i.e. the DBS Central Processes.

Child In a hierarchical structure, denotes an immediate descendant of a given component. A child is thus located one hierarchical level below its **parent.**

Client A process or program that makes use of services provided by another process or program (called service provider or **server**).

Command Elementary instruction or statement from which a functionality is called.

Commonality Commonality is the software characteristic that ensures the use of interface standards for protocols, routines, and data representations.

Communication Session A communication session is a logical exchange of messages between two users of the network software, such as in a response/request scenario when a request from user A solicits one or more responses from user B. The session can be considered to begin when the request is sent and finished when the last response has been received. This definition of a session is only valid when talking in terms of the network software. It should not be confused with an OSI session because OSI has its own widely accepted definition.

Compilation Unit Smallest unit of code that is accepted by the compiler. In UCL, there are 3 types of Compilation Units: Automated Procedure (AP), Library Specification, and Library Implementation (or Library body).

Component Component is a generic term used to cover any item in the higher levels of the software architecture (i.e. product, assembly and subsystem).

Composite Function Block In CSS, atomic function blocks may be composed together with parameter blocks and other composite fucntion blocks to a new unit, called composite function block.

Condition In CGS, Condition has a specific meaning: It is an attribute of a measurement/software variable or derived value. It specifies, when another measurement is enabled for processing, an AP is started or a limit set is switched, in terms of a specific value of the item it is defined for.

Configuration Control Unit (CCU)
A CCU is a Configuration Unit used to define and control other Configuration Units in the DB. It identifies which specific combination of CDU instances make up a particular configuration. A CCU may, in turn, point to lower level control units, thus leading to an hierarchical configuration tree whose topmost (root) component corresponds to the overall Element Configuration. CCU can be described as directories of items (CDU) in the DB.

Configuration Data Unit (CDU)
CDUs are composite entities containing the actual data items

Copyright per DIN 34

	(grouped into individual units for configuration management purposes). CDUs contain the actual data items, while CCUs are collections of references to CDUs only.
Configuration File	A disc file which a software component reads during process start up to retrieve configurable parameter. A configuration file is an ASCII data file and may be changed via a standard editor.
Configuration Unit (CU)	Collection of MDB items treated as a single unit for configuration management purposes. CUs are of two kinds: (a) Configuration **Data** Units (CDU), which contain the actual data (b) Configuration **Control** Units (CCU), which contain reference information (CU name, version number, etc.) about other CUs, just like a directory in a file system.
Configuration Data Item	All Onboard Data shall be constructed as CDI's.
Configuration Item (CI)	An item, which defines a software configuration, that for the purposes of configuration management is required to be considered as a single entity.
Configuration Management	The control and coordination of the development of a system.
Consistency	Consistency is the software characteristic that ensures uniform design and implementation techniques and notations. In the context of the mission dabase, consistency means valid data with defined references and allowed values.
Consistency state	CDUs or CCUs have a defined consistency state: LOCAL VALID : for a CDU: The data is consistent within the CDU scope LOCAL INVALID: for a CDU: The data is not consistent within the CDU scope, but may be consistent wthin a CCU scope. GLOBAL VALID: the data is consistent within a CCU scope.
Correctness	Correctness is the degree to which the software component satisfies the specified requirements.
Crew Procedure Language (CPL)	In COLUMBUS, this a special language used onboard for secification of crew actions

D

DATA_API	A view based interface to the Mission Database. Provides views for retrieval from the Mission Database, functions for setting the configuration and functions for commit/rollback within the chosen configuration.
Database	A common or integrated collection of interrelated data whose purpose is to serve one or more applications.
Database Integrity	Refers to the state in which the database is considered to be undamaged (both physically and logically).

Copyright per DIN 34

Database Management System The software responsible for the actual definition, storage and manipulation of data in a Database at both the physical and logical level.

Database Administrator (DBA) The person(s) responsible for the operation and maintenance of a DBMS. Specific user in Oracle with special access rights.

Data Dispatch The process of supplying data from one instance of CGS/TES to another instance of CGS/TES or CGS/HCI

Data Entry / Data Maintenance

Generally refers to the process of entering and/or updating data in the database.
In this context, the term ”maintain” refers to any operation which alters the state of the Database, i.e. add (insert) new data, modify existing data, or delete data.

Data Interface SAS A special type of special appliation Software that can handle ADUs and GDUs under the control of TES. This type of SAS constitutes the Software interface between CGS and the front end equipments that measure data, acquire telemetry, send stimuly or TC packets to the unit under test.

Data Processing SAS A special type of special appliation Software that performs a special data processing. This type of SAS is not controlled by TES but on its own takes the initiative to read data from TES or send data to TES for further processing.

Data Set TEV generates from archive or from the engieering value log extracted and converted (calibrated) values and stores them into objects called ”Data Set”. Data Sets can then be converted to Data Listings, Graphs or Statistics or may be further processed by user specific programs (SAS).

DB Server In CGS, a database server node is a logical node which provides services to store and manage access to the Configuration DB as well as to the Test Result DB.
A logical DB Server node has to be mapped to a physical processor where it runs on (DB server processor).
The DB Server node normally is allocated to the CGS Server, which is the network node executing the Oracle System and acts for the other nodes as file server.

Deadlock Situation in which two or more user processes cannot complete their transactions because each process is holding a resource that the other process requires in order to complete.

Decalibration Decalibration means calculation of values for enditems according to predefined calibration curves, from an engieering format into a raw format. The result of a decalibrated value is the Raw Value, which can be sent to devices or can be simulated or is suitable to be stored into CCSDS packets.
Calibration is the reverse process (–> Calibration)

Default A value supplied by the system when a user does not specify a required parameter, qualifier, or attribute.

Copyright per DIN 34

Derived Value	Derived Values are specific enditems whose values are calculated from other enditem's values. An UCL expression can be used to specify the calculation.
Display (or Screen Display)	In this context, refers to an area on the physical screen surface assigned to applications for the purpose of communicating with the human users. It may comprise one or several individual partitions (windows) each assigned to a different application.
Distributed Database	A collection of databases that can be operated and managed separately and also share information.

E

End Item	A part of the EGSE or Spacecraft (UUT) that can be addressed by the S/W resp. the user and cannot be broken down into lower level items. Examples are: sensors, actuators, S/W variables, measurements,telecommands, automated procedures. Enditems are addressed via the Columbus name tree (pathname). Note: When discussing DB structures, the term enditem is sometimes used to refer to any DB entry, differentiating from 'views' to database tables or from virtual trees.
End Item Mapping	In the Mission DB, user defined enditems may exist. These enditems may have all new aggregates/attributes or may partly be mapped to existing enditems and thus share their attributes. The mapped attributes are visible to the user as any new attribute of the enditem. When handled in CGS, the type of the enditem might be handled as the new, user defined type, or may implicitly be handled as the mapped, original CGS type. In this way, user defined enditems may easily extend the existing enditem types by preserving the existing procedures.
Engineering Unit	The Engineering Unit defines the units of measure for a Engineering Value. Refer also to 'Calibration'
Engineering Value	The Engineering Value is the result of calibrating raw values. Refer to 'Calibration'
Engineering Value Log	In CGS, enditem values may be logged (stored with a time tag) in calibrated format in special logbooks as part of the Test Result DB. The values may be evaluated in an offline session without need for re–calibration.
Environment Variable	A UNIX term describing variables whose values are known to a process and definable via UNIX commands. They are indicated by a "$" prefix. In CGS, environment variables are used to set debug environments for processes and as a shortcut for disk/file pathnames.
Error Message	The collection of data (textual description, attributes such as reference number, criticality, time of occurrence, etc.) which describes the error which has occurred.

Copyright per DIN 34

Evaluation Definition	A file containing a definition used by TEV. Following types of evaluation definition are defined: – Selection Criteria (for logging event evaluation) – Statistic Definition (for statistical analysis) – Listing Definition (for listings) – Graph Definition (for graphical presentation of values) – Report Definition (for generation of a test report).
Evaluation Result File	A file containing the output of a TEV tool for further processing or for storage.
Evaluation Session	A session in the TRDB that contains results generated during data evaluation, i.e. using TEV Evaluation Sessions may be further extended by adding additional evaluation results. Evalution Sessions may be stored online in the TRDB or exported to final archive devices.
Event Log	In CGS, events (errors, out–of–limit exceptions, user input etc.) may be logged (stored with a time tag) as part of a test session within the Test Result DB. The events may be evaluated in an off-line session by producing a selected list of events, ordered according to time tags. Events mainly consists of messages generated by CGS due to any kind of activity, including the execution of specific event–generating UCL statements.
Exception	An out–of–limits condition or a status inconsistent with the value obtained by the monitoring function. In both cases, the current operational state is taken into account. An exception may also be derived from other values. Internally in CGS, the term exceptions is also refering to Ada exceptions, which are defined error states of the software.
Execution Session	A session in the TRDB that contains results generated during online test activities. Execution Sessions are opened at the beginning of a test and closed at the end. All archived and logged data is stored in this session. Execution Sessions may be evaluated by TEV during or after a test. The evluation results are stored in –> Evaluation Sessions. Execution Sessions may be stored online in the TRDB or exported to final archive devices.
Export	In the MDB context, this term refers to the process of extracting data from a DB and preparing it for inclusion (**import**) into another MDB instance. In the TRDB context, this term refers to the process of extracting test session data from a TRDB and preparing it for inclusion (**import**) into another TRDB instance.

F

Copyright per DIN 34

Fault An accidental condition that causes a functional unit to fail to perform its required functions. A fault if encountered, may cause a failure.

Final Archive Test Session data (Files, TRDB contents) is transfered to disks during the ongoing tests. This process is called ”final archiving”.

Flexibility Flexibility is the extent of effort required to change (modify existing) software to accommodate changes in requirements.

Flight Software All developed or procured software which will execute in the flight configuration after launch. The applicability of requirements in this document to flight software is normally limited to flight software which forms part of the Space segment development. Sometimes the term onboard software is used to mean the same.

Formal Document A document that is released as part of a CI instantiation.

Formal Language Specification Formal language specifications are done in extended Backus–Naur Form (EBNF).

Formal Software A piece of software that is released as part of a CI instantiation.

Function Block A component of the Simulator (CSS) functionality which is invoked as a single entity, c.f. Base Function and Function Mask.

Function Mask A mask which provides a map onto the functions provided by the function block, implying the selection or non–selection of each function, c.f. Base Function and Function Mask.

G

Generation Data Unit (GDU) An GDU is a data unit that contains data generated by a testnode and sent to an SAS. In general, GDUs contain commands/requests or stimuli (i.e. requests for analog / digital output). GDUs may contain as well a complete TC packet.

Ground Data Item All Ground Data shall be constructed as GDI’s. See document NO TAG for details.

Ground Software All software that executes in any ground computer or in the flight configuration computers during pre–launch ground operations.

Ground SWEU All COLUMBUS ground software is configured into ground SWEUs. Each ground SWEU is any software unit or component which can be replaced as a single item. As such a ground SWEU is primarily a configuration management item.

H

Heterogeneous Environment Refers to a system comprising different types of processors or operating systems.

Hierarchical Name Tree see **Name Tree**

High Level Command Language (HLCL)
HLCL comprises a set of commands that can be given either inter-

Copyright per DIN 34

actively as HLCL command or in a predefined HLCL Sequence. In VICOS the language is used for control of the online test execution as well as for test system setup and test evaluation.

HLCL Command A HLCL command is any command which can be given interactively by the user. HLCL commands can be given in specific command windows or from Synoptics or from a HLCL command sequence. Some of them are translated to UCL interactive commands and transfered to the UCL Interpreter running on the EGSE test nodes.

Home Directory The directory location assigned to a particular application of CGS, also called Application Home.

Homogeneous system Refers to a system in which all processors are of the same type or family (usually from one vendor).

Housekeeping Value In CGS, some internal values are maintained during chkout operations. The values provide for status information on CGS processes and loaded data. They are referred to as Housekeeping Values.

I

Import In the MDB context, this term refers to the process of receiving or including data from an external (possibly remote) DB into the local DB.

In the TRDB context, it means retrieving a test session from an external device (magneto optic device) into the TRDB and make it available for evaluation.

Integrity Integrity is the extent to which the software component controls access to system resources. resources here include database items, functions, and software controlled hardware.

Independence Independence is the software characteristic that ensures that it the software does not depend on its environment (e.g. the computing system, operating system, utilities, I/O routines, and libraries)

Instance In CGS, the instance of a process resp. a logical node is the actual process/node running on a physical processor. Instances are named with their logical process name (DBS,HCI,TES,TSCV,CSS,TEV) and an instance suffix ”_01” .. ”_32”. E.g. for the different test nodes within a system, 3 might be defined: TES_01, TES_05 and TES_10. Each instance is running on a different machine (test node). Instance Names are mapped to test node’s pathnames using the EGSE–NODE enditem type in the MDB.
Instance Names are defined withn the System Topology Table, where the mapping to physical processor names (host names) is specified.

Interoperability Interoperability is the extent of effort required to facilitate the interface of one software component with other systems or software components.

Copyright per DIN 34

J – L

Level Name The name which identifies one node at a particular level in the MDB hierarchy. A long path name is a concatenation of level names.

Library see Symbol Library**,** UCL Library

Limit In CGS, monitoring is driven by limits defined for each enditem. Lower and upper limits can be defined. When CGS detects an enditem value which is 'out–of–limits', a message ('exception') is raised if a predefined 'count' of limit violations is reached. Together with the generation of exceptions, automatic actions (start of AP, generation of stimulli/TC) may be initiated. 'Limits' for discrete or digiital enditems are referred to as 'Expected Values'
There are two kinds of limits: 'hard (danger) limits' and 'soft (nominal) limits'. Hard limits cannot be changed online and are checked with higher priority than soft limits. Soft limits can be changed by the test operator in an ongoing test (online).
Each enditem may have a set of soft limits defined.

Local Time (LT) The time to which the system clocks are set is the LT. It can be e.g. the official local time, in use at the location of the EGSE's installation or GMT or UTC,
The time server for LT will be the time of a dedicated clock (NTP Master). In the EGSE this is normally the MTP system clock or an external clock (Master Time Unit).
In CGS all local clocks are synchronised using the NTP software.

Locking Mutual exclusion mechanism used for controlling concurrent access by multiple users/applications to a shared resource.

Logging See 'Event Log' and 'Engineering Value Log'

M

Maintainability Maintainability is the extent of effort required to find and fix errors in the software component.

MDB Item, MDB Object These two terms are used interchangeably to denote a uniquely identifiable entity that has been defined in the Mission Database. An MDB Object or Item may be decomposed into lower–level items according to the hierarchical nametree conventions, see **Nametree** below.
An **End–Item** is an MDB item located at the lowest hierarchical level (leaf or terminal node), and hence cannot be further decomposed.

Master Archive Refer to 'Test Result DB'

Master Test Processor (MTP) One of the test nodes in the Test Configuration is playing always the role of themaster: Some general functions are only executed on this node:
– Fetching Housekeeping Values from general services

Copyright per DIN 34

– Archiving SMT setup commands
– Executing the Overall Setup AP started by TSCV
etc.

Measurement A measurement is a single (analog or digital) intput from a (measurement) device. A measurement may be received by CGS in a TM packet or any other ADU via SAS.
Note: in CGS, sometimes the term measurement is used for all enditems describing engneering values: Software Variables, Derived Values and the measurements as described above.

Mission The performance of a coherent set of investigation or operations in space to achieve space programme goals. A single mission may require more than one flight, and more than one mission may be accomplished on a single flight.

Mission DB (MDB) The Mission Database contains configuration descriptions for the UUT (Columbus Subsystems and Elements) as well as for the EGSE. It contains all definitions describing the UUT, the EGSE configuration and the test to be executed.
It is the central repository for all HW / SW configuration information about Flight Elements, Payloads and associated Ground Support Equipment.

Model Configuration Model Configuration represents a number of Model Functions complete for conversion into an Executable Model Image indepedent of the level of breakdown.

Model Function Model Function represents one or several functions to be simulated on different levels of decomposition. A top level Model Function will be broken down via several levels of decomposition to the lowest level Model Function presentation containing functional definitions coded in a subset of ADA or represented in decision tables.

Model Image The Model Image represents compiled and linked code and data of a Model Configuration ready being loaded and executed.

Modularity Modularity is the characteristic of software that ensures a highly cohesive component structure with optimum coupling.

Monitoring Refer to –> Limit

N

Nametree Hierarchical (tree) structure within the MDB which portrays the hierarchical decomposition of Flight and Ground Configurations into systems, subsystems, equipment, etc. The topmost node of the nametree (called the root node) might designate the Flight Configuration, whereas terminal nodes (leaf nodes) represent the items that cannot (or need not) be further decomposed, i.e. the so–called **end–items**.

Copyright per DIN 34

Each MDB object is thus identifiable by a **pathname** indicating the succession of nodes to be traversed to reach that particular item in the Nametree.

Network Time Protocol (NTP) NTP is a time synchronisation protocol used to minimise the offsets between the system clocks of the computers in a network. In this protocol the NTP–clients periodically request time information from the NTP–server. Based upon this information, the NTP–clients adjust their own clocks in order to get the offset w.r.t. the NTP–server to the minimum.

Network A group of computers (workstations) and/or terminals that are linked together to allow the sharing of resources (data and peripherals).

Network Information System (NIS)
A system for management and setup of several UNIX nodes and user in a network. Part of the Solaris Operating System.

Node Physical Node: Any computer within a network.
Logical Node: Any CGS 'function' that may be distributed to a physical node

Notice window Open look terminology. A window which pops up to display any kind of message. The user input focus is bounded to the window. The user has to select a button (e.g. "continue") to confirm, that the message has bee recognized.

O

Onboard Software All software that executes within a flight configuration during on–orbit operations.

Online Test Control (HCI) This software package resp. process provides the main user interface for online testing. It runs on each workstations and must be startet by each user via the task selector.

Operability Operability is the ease by which a person can use a system comprising software and hardware.

Operating System The system software that controls the computer and its parts, performing the basic tasks such as allocating memory, and allowing computer components to communicate.

Operator The person who operates any system comprising software and hardware using the provided computer input devices (e.g. keyboard, pointing device, voice input, pushbuttons, switches etc) and/or computer output devices (e.g. screen, printer, lamps etc).

P

Parent In a hierarchical structure, denotes an immediate ancestor of a given component.

Pathname A pathname identifies in a unique way an enditem.
A pathname is structured hierarchically according to the name tree

Copyright per DIN 34

defined in the Mission DB.

Portability Portability is the extent of effort required to transfer the software component from one hardware or software system environment to another.

Procedural Interface A procedural interface is an interface which is implemented by a set of procedures, usually grouped into an Ada package(s). The interface may be included in the interfacing component without concern for the underlying implementation, and so allow development and testing of the underlying component.

Product Tree A tree structure that defines the constituent CI instantiation for a particular development.

Protocol Rules and conventions for organizing data to be sent from one machine to another. The protocol enables the destination machine to recognize that the data is addressed to it, check the data to make sure that it is valid, unpack and decode the data, etc.

Q

– No Definition –

R

Realtime Node Either a Test Node, Master Test Processor or a Main Computer System

Reconfigurability Reconfigurability is the characteristic of software that ensures continuity of system operation when one or more processors, storage units, or communication links fail.

Recovery This is the process where a Database which is damaged (or assumed to be so) is restored to a previous state known to be consistent.
In CGS, recovery scripts are provided that allow for restorage of DB contents, especially for the TRDB.

Reliability Reliability is the extent to which the software component consistently performs the specified functions or any interface requirements.

Replay Session CGS allows to retrieve all data stored in archive files of an execution session and display and process the data in the same way as done during the online session. This mode of operation is called Replay, and the execution session foreseen to be replayed, is referred to as Replay Session.

Report In the context of this document, a report may be defined as any human–readable description of one or more MDB items. It is an assorted collection of information usually presented to the user in form of a table or itemized list (tabular format).

Copyright per DIN 34

	A report's specification contains the instructions for generating the report, e.g. data selection criteria, formatting instructions, and sort order. On request, a report is generated, i.e. the predefined instructions are executed, and the resulting output routed either to the workstation's screen (on–screen report), to the printer or to a user–selected file.
Resource	Any of the component parts of the System, or the facilities that it offers (e.g. power, communication channels, etc.).
Response time	For interactive transactions, this refers to the time elapsing between the start event (e.g. pressing the ENTER, ACCEPT, or COMMIT key) and a response event (e.g. the first character of the reply reaching the user's terminal).
Reusability	Reusability is the extent of effort required to convert a portion of the software component for use in another application.

S

Safety	Safety is the absence of hazardous conditions.
Shell	The UNIX Shell is the means by which direct access to the UNIX operating system is enabled. Types of UNIX shell such as 'csh', (C Shell), 'ksh' (Korn Shell) or 'sh' (Bourne Shell) exist within which UNIX commands and software programs executed etc. Software programs can be written in the 'shell' language, which can then be executed within the Shell.
Short Identification (SID)	A unique identification value (integer) for an end item in the Columbus Name Tree, which is used internally in CGS to access that item.
Simulated Mission Time (SMT)	Mission Time starts counting the moment a spececraft is launched or enabled. In order to perform the required EGSE tests, this time can be simulated, and the possibility is provided to manipulate SMT, i.e. to set, stop and continue SMT. In CGS, SMT is maintained by the Time Server Process (TSP) and may be controlled via UCL statements. SMT is distributed to all nodes. EGSE may further distribute the SMT to frontends and finally to the units driving the onboard time.
SMT domain	A set of netork nodes that share the same SMT. In CGS, several SMT domains may coexist. CGS allows to define up to 5 Test Confugurations in parallel, each having a different SMT domain.
Software Variable	An enditem describing an engineering value, which is generated by software within CGS or within APs/SAS. A software variable may be bound to an Housekeeping Value of CGS. It then allows to get read access to these values.
Spawn	Term to describe the initiation of an executable piece of software eg running under the UNIX operating system. It will create a stan-

Copyright per DIN 34

dalone program running as a UNIX process. A process can be spawned from another application or directly from UNIX.

Specific Applic. S/W (SAS) Specific Application S/W (SAS) is the S/W executed in parallel with CGS/TES on the same H/W. It is written in Ada and executed under the same OS as CGS. It is linked to CGS services using VI-COS supplied interface libraries. It serves for data delivery to CGS, reception of commands/stimuli and specific processing of data.

Stimulus A stimulus is a single (analog or digital) output to a (stimuli) device.

S/W Exchangeable Unit All COLUMBUS ground software is configured as SWEU's. Each ground SWRU is any software unit or component which can be exchanged as a single item. As such a SWEU is primarily a configuration management item.

S/W Replaceable Unit All COLUMBUS flight software is configured as a set of SWRU's. Some flight SWRUs can be replaced online. Otheres are replaced by reloading the pocessor. All flight SWRU's are configuration management items in the same way as SWEU's.

Symptom Indicator of faults or failures.

Synoptic Displays Synoptic Displays is a CGS window service which allow the human user to display and manipulate Synoptic Picture within windows at the workstation screens.

Synoptic Picture A Synoptic Picture is a predefined graphical picture that represents the physical devices and subsystems (EGSE or UUT). It contains output elements that are dynamicly animated (to indicate status / values) as well as input elements that allow for commanding the devices/subsystems by direct mouse manipulation.

System Administrator A person responsible for the operation and maintenance of the **operating system** of a computer. In CGS, the system administrator may or may not be the same person as the **CGS Administrator**.

System Tree The toplevel tree elements within the name tree are defined within the MDB as the System Tree. It may exists in several versions within the same Mission of the MDB.

System Topology Table The system topology table contains the mapping ' Logical Node Name (Instance Name) to Physical Host Name' of all possible nodes for an EGSE. The logical node name is used by CGS always, while the underlying services (e.g. UNIX) use the physical host names to identifiy the processors in a network. The physical host names must be defined in the UNIX /etc/hosts file.

T

Telecommand (TC) The data uplinked from the ground to a spacecraft is known as telecommands. It contains requests to the onboard system or bulk data. In the Columbus EGSE the data is sent to the Spacecraft via the

Copyright per DIN 34

	TM/TC Front End. The protocols used are baselined to be CCSDS path service, and therefore TC data is uplinked in CCSDS packets. In CGS, a TC is always a single request. Bulk data is to be uplinked by S/W outside CGS (SAS) using CGS services. TC and Stimuli are handled in a similar way. TC packets are transfered from CGS to SAS inside a Generation Data Unit (GDU).
Telemetry (TM)	The data downlinked from a spacecraft to the ground is known as telemetry data. It contains status data of the onboard system. In the Columbus EGSE the data is acquired from the Spacecraft via the TM/TC Front End. The protocols used are baselined to be CCSDS path service, and therefore TM data is downlinked in CCSDS packets. In CGS TM packets and measurements are handled in a similar way. TM packets are transfered to CGS inside an Acquistion Data Unit (ADU).
Test Configuration	A test configuration describes the EGSE processing configuration needed for a (or a set of) specific test(s). It defines the participating nodes, the allocation of functions/data to nodes as well as the SAS to be used/distributed on that nodes. In CGS, the user may select a test configuration defined in the DB before the EGSE is setup. CGS will automaticly setup the system according to the test configuration selected.
Test Node	In CGS, a test node is a specific logical node which provides the data interfaces to the UUT, either directly or via frontend equipment. Test nodes require data from the UUT, generate data to it, process data (calibration and monitoring) and interprete Automatic Procedures. A logical test node has to be mapped to a physical processor where it runs on (test processor).
Test Results	In VICOS, the term test results is summarising all data produced during a test session that are foreseen for long term storage or offline evaluation. Test Results comprise at least: – raw data archive files – event logging data – engineering value logging data – printable reports – graphical output – data sets produced for storage of enditem values
Test Result DB	The Test Result DB (TRDB) contains all data (test results) produced in different test sessions, either during test execution or test evaluation It contains an index of all data items (Master Archive), indicating the name, the related time frame and the location of storage for each item. The Test Result DB is managed by DBS and consists of Oracle Tables as well as of OS files.

Copyright per DIN 34

The Master Archive is a sort of index that allows to retrieve data stored on the Mass Storage according to criteria like time, test session, evaluation session.

The Mass Storage contains physically all the data: Logbooks, Archive Files, Evaluation Result Files. It is accessed thru the Master Archive. The data will be stored temporarily on magnetic disc and then on several storage media (Long Term Storage Medium e.g. optical disc).

The visible part of the TRDB (”Online Data”) is the part that can be directly accessed at a certain instant of time. That means, it is the Master Archive plus the data of the Mass Storage stored on magnetic disc and on the Long Term Storage Media actually mounted.

Test Session

A test session is a term identifying a sequence of operations and activities within a given time frame, related to a specific set of test objectives, together with the generated test results and a reference to the used configuration. A test session has a unique name, and an index of all results produced in the test session is stored in the Test Session Table within the Master Archive as part of the Test Result DB.
There are two kinds of test sessions:
– the Online Test Session,
which is produced during an online test
– the Evaluation Test Session,
which is produced by a specific user when data evaluation is performed offline.

Time Services (TSS)

In CGS, the maintenance of the Simualted Mission Time (SMT) is provided by a software product called Time Serives Software (TSS). It establishes a time erver process on each CGS node.TSS includes also the setup scripts for the Network Time Protocol (NTP).

Trace

A trace provides a link between two different stages in the development lifecycle in order to provide the traceability for a development.

Traceability

Traceability is the characteristic that provides a thread of origin and a thread of implementation. The thread of origin (or the reason for existence) is from the implementation to the requirements with respect to the specified development envelope and operational environment. The thread of implementation is in the opposite direction and is used for verification purposes.

Trace Object

A trace object is an object from one stage of the software development that has been linked by a trace to another object at another stage of the development (e.g. requirement, HOOD object, test procedure, etc).

Copyright per DIN 34

Transaction	Single thread of related activities representing a sequence of operations initiated by the occurrence of a stimulus and ending with the required response.

U

UCL Library	Encapsulation mechanism for UCL data structures and operations. (can be viewed as a collection of functions, procedures, types, etc.) A UCL Library corresponds to a *package* in Ada or a *module* in Modula–2.
Unit Under Test (UUT)	UUT means a spacecraft or a part of a spacecraft where the EGSE is connected to and which is foreseen to be checked out using the EGSE and CGS.
User Control Language (UCL)	Columbus Test and Operations language (used for real–time control & monitoring purposes in both the onboard and ground environment)
User	Throughout this document the term **User** refers to any person using CGS–provided services. Users are grouped into different classes or categories and will be assigned different privileges based on the task they perform.
User Event	A specific log item that may be used to identify points within the test activities where evaluation can refer to in a logical way. User events are the only events that have the log type UEVT. InTEV, a time frame may defined according to user events, which are always selectable to the user. In UCL, a user event is generated when calling the procedure USER_EVENT.
User Working Environment	A user working environment is a specific S/W environment, which is appropriate to a human user (e.g. test conductor, test observer) and his task. It enables the user to start automatically the applications with the corresponding windows needed for his work (e.g. creation of a Synoptic Displays window and start of the AP's belonging to it). See also 'S/W Environment (VICOS User)'.

V

Version	In the course of its life cycle, a Configuration Unit (CU) usually undergoes several modifications due to evolving user requirements, design changes, etc. It will thus possibly exist within the MDB in many different forms or instances (CU occurrences) commonly referred to as *versions*, e.g. DMS Version 3.2.1. In the Configuration Management (CM) context, however, the various CU occurrences. are classified according to the types of changes that have been made. The terms **versions**, **issues**, and **revisions** are then used to differentiate between the following 3 cases:

Copyright per DIN 34

– Modifications due to *requirements changes* which result in a new **version**
– Modifications due to *design changes* which result in a new **issue**.
– Modifications due to *bug fixes*, *repairs* or *other corrections* (affecting neither the design nor the requirements) which result in a new **revision**.

(In the above example, the CU Identifier ”DMS Version 3.2.1”, therefore, refers to Version 3, Issue 2, Revision 1 of the DMS)

Version_Id_Table The version_id_table is the storage area outside the configuration data base, where the information is stored, which special application S/W (object and data files) and which CGS version is stored/ installed .

VICOS CGS is mainly divided into the Simulation Services (CSS), the Mission DB (MDB/MDA) and the checkout services (VICOS). The term VICOS is outdated, but sometimes still used to describe the checkout services of CGS.

Virtual node A virtual node is an item of the Mission Database that corresponds to an incomplete pathname, that means the attributes of the item are child items and not real attributes like measurement definition etc.

W

Widget An object providing a user interface abstraction (for example, a Scrollbar widget).

Working Directory A disc space containing the intermediate files for a specific tool and a specific user.

Workstation Node A workstation node is a logical node which provides the interface to the operator and executes operator related functions for test execution monitoring and control, for test setup, for test preparation and for test evaluation. It runs the CGS HCI, the TSCV and the TEV or the USer Interface of the Simulation (MOCS)
A logical workstation node has to be mapped to a physical processor where it runs on (workstation processor).

X – Z

X11 X Window System Version 11 is a library which provides functions to write applications with a graphical user interface running on a network.

Copyright per DIN 34

C END ITEM TYPES

See chapter 6.6 and the related ICD's

Copyright per DIN 34

D CGS ERROR MESSAGES

This chapter provides a collection of all possible error messages from the CGS tools and tips and hints how the user can take remedial actions.
Note that all error report mechanism which do not conform with the error message handling as described in chapter NO TAG will be described here in detail.

D-1 Commercial Tools

D-1.1 The ORACLE Database

All ORACLE errors are documented in the ORACLE user documentation provided by the vendor.

D-1.1.1 ORACLE On–Line Help Facility

Additionally there is an on–line help facility provided by the ORACLE system. A typical ORACLE error message is shown in Figure 1. The error message starts with a three–letter prefix which identifies the facility. In the example you see that ”ORA” is the facility which reported the error and ”01017” is the error number.

Make sure that the file ..../oracle.../bin/oerr is visible on your workstation.

```
marlies@csf_12: sqlplus

SQL*Plus: Version 3.0.12.4.1 - Production on Thu Jun 29 13:33:38 1995

Copyright (c) Oracle Corporation 1979, 1992.  All rights reserved.

Enter user-name: cssaiv
Enter password:
ERROR: ORA-01017: invalid username/password; logon denied
```

Figure 1 : *Typical ORACLE Error Message*

To get more information about the error open a window and type **oerr** <facility> <error–number> (see the example Figure 2) An explanatory text will be displayed beside the error number. A description of the cause follows in the next line. At last there are suggested actions to resolve the error condition.
Note that there are two types of suggested actions:

– DBA – actions to be performed by the Data Base Administration
(user needs special knowledge and/or special privileges)

– User – this is what the ordinary user can do

```
marlies@csf_12: oerr ora 60
00060, 00000, "deadlock detected while waiting for resource"
// *Cause:  Transactions deadlock one another waiting for resources
// *Action: DBA - Look at the trace file to see the transactions and resources
//                involved
//          User - retry if necessary
```

Figure 2 : *ORACLE On–Line Error Help*

Copyright per DIN 34

Figure 3 shows an error message issued by I_MDB. The text is originated by I_MDB, the ORACLE error number is added, the facility is omitted.

- *To get more information about oracle errors reported by I_MDB use the default facility ”ora” and the error number.*
  Note that the ”–” sign standing before the error number has to be neglected to get the correct error message.

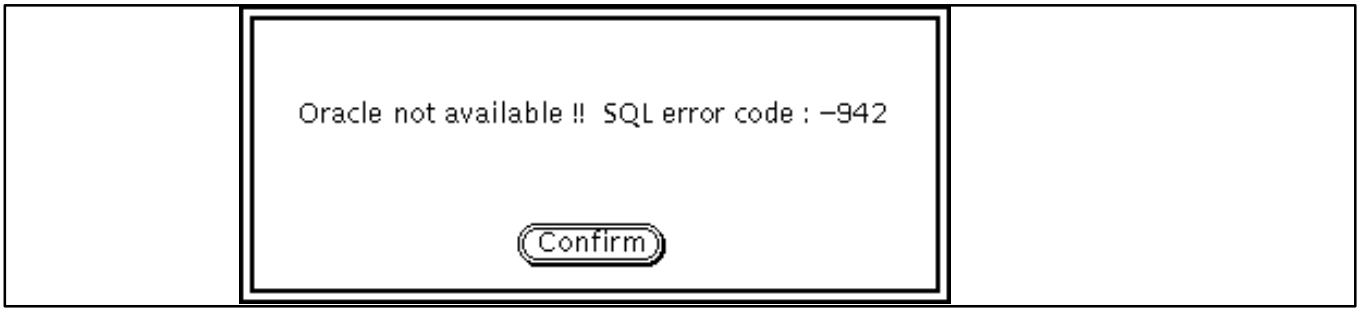


Figure 3 : *Oracle Error Number Added to an I_MDB Specific Error Message*

D-1.2 The Ada Compiler

- *See Manuals of the delivered Ada compiler for descrciption of complier related error messages.*

Copyright per DIN 34

D-2 Test Preparation

D-2.1 MDA Error Messages

D-2.1.1 Consistency Checker Error Messages

For consistency checker error messages refer to the Appendix N.

D-2.1.2 Export/import Error Messages

For export/import error messages refer to the MDA Reference Manual (Ref 2.2.1.3)

D-2.1.3 Batch data entry Error Messages

For BDE error messages refer to the MDA Reference Manual (Ref 2.2.1.3)

D-2.1.4 I_MDB Error Messages

For I_MDB error messages refer to the MDA Reference Manual (Ref 2.2.1.3)

D-2.1.5 Generate SCOE Files

For Generate SCOE Files are the same data checks valid as for the consistency checker.

Copyright per DIN 34

D-2.1.6 GWDU: Ground Synoptic Display Editor

For GWDU error messages refer to the GWDU User Manual (Ref Doc. 2.2.2.1)

D-2.1.7 CLS Editor and Compiler

The *Columbus Language System (CLS)* comprises several language related software components for UCL, HLCL and CPL.

Automated Procedures are software programs written in the User Control Language (UCL).

UCL programs are edited and compiled off–line. During the compilation process, the UCL code is transformed into a binary intermediate code which is later executed (*interpreted*) in the target environment by a dedicated program (*interpreter*).

In addition, the compiler generates:

1. a *symbol table* which is internally used by the compiler
2. a *debug table* intended for the source–level debugger
3. an MDB *cross–reference list* intended for the Consistency Checker of the MPS.

In case of UCL programs full checks of availability and compatibility of accessed resources are performed at mission preparation time.

During a UCL editing session errors found by the compiler are marked with a dark rectangle in the editor field.. Additionally the error messages are displayed in a text field in the lower part of the window (see Figure 4). The error messages are written in plain and detailed text. The errors messages are self–explanatory, therefore a list of possible syntax errors is not provided.

Additionally the user gets support from the on–line Syntax Help. The different syntax rules can be checked by pressing the **Help** button (see Figure 5) and the user can copy the desired language construct from the on–line Syntax Help (by pressing the copy button), an easy way to avoid syntax errors.

Internal errors (i.e. errors in the CGS software), which may occur, are clearly marked as internal errors. There are no corrective actions recommended to be performed by the user. The internal error has to be reported to the CGS maintenance team.

If an internal error message is displayed, write a SPR (software problem report) immediately and report the error to the maintenance team.

Copyright per DIN 34

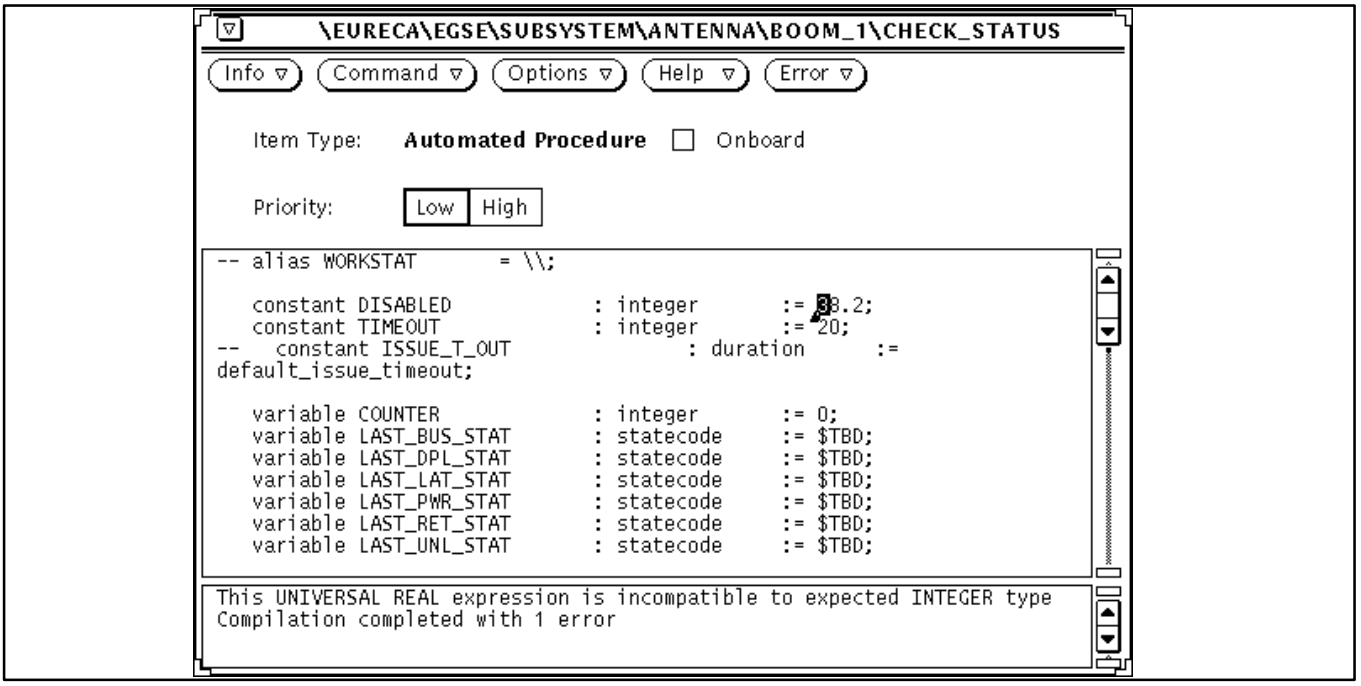


Figure 4 : *The UCL Compiler Error Message is displayed in the lower part of the editor window.*

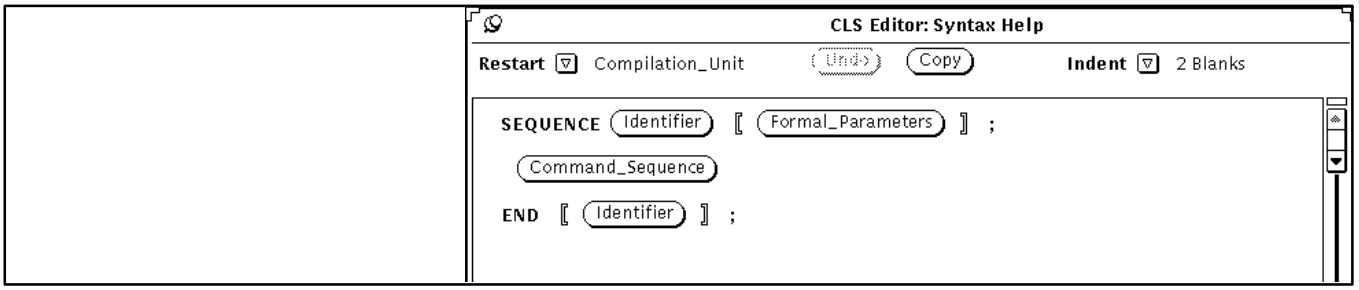


Figure 5 : *The On–Line Syntax Help shows the UCL syntax*

D-2.2 HLCL Command Sequences

HLCL is an interactive command language used in different environments: within an interactive session the user types comands which are executed immediately by the HLCL command interpreter.

HLCL command sequences are edited and compiled off–line. The error handling mechanism is just the same as provided with the UCL editor. Also the Syntax Help function is available for HLCL.

Before the HLCL command sequence can be stored in the database a syntax check will be performed. Errors are reported in the same way as descrbed in section 5.2.5 (automatec procedures). Figure 6 shows the erronous HLCL sequence.

Copyright per DIN 34

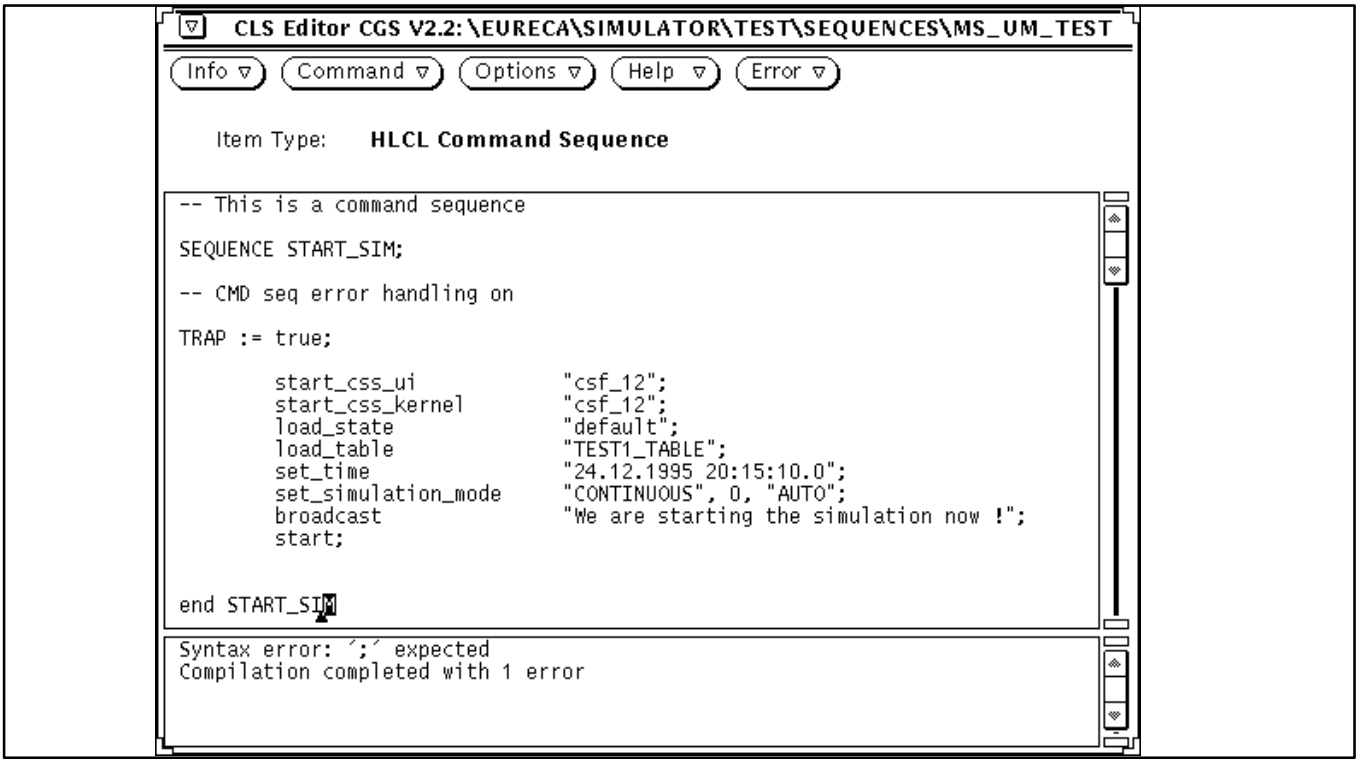


Figure 6 : *A Syntax Error is Indicated by a Dark Rectangle, the Error Message is Displayed in the Lower Part of the Editor Window.*

The syntax of the HLCL language written in a command sequence is *not* completely equal to the way HLCL is used as interactive language.

Within the command sequence the same syntax rules apply as in a session, but some session specific relaxations and extensions are not valid:

- Abbreviations are not allowed.
- The end of a line is not a command terminator, i.e. commands may be formatted over several lines if desired but must be terminated with a semicolon (in any case).
- Engineering units must not be omitted.

Copyright per DIN 34

D-2.2.1 Model Development

D-2.2.1.1 General

The following section gives a short overview about the different error handling methods used in CSS.

- *Note that only run time errors which occur during model execution are delivered to the central CGSI error handler.*

Depending on the action the user is currently performing (model development, model compilation, model observation and model execution) there are different ways to display errors and warnings.

- *Note that different parts of CSS use different error handling methods.*

D-2.2.1.2 Error messages during model editing

The Composite Editor window has a separate message line at the top of the editor area.
During model editing information and error messages are displayed in the message line.

- *Not all information displayed in the message line are error messages !*

Most error messages give also some advice to take corrective actions (see Figure 7)

While editing a decision table or AIL code error messages are displayed in the message line of the appropriate Atomic Editor subwindow. (see Figure 8)

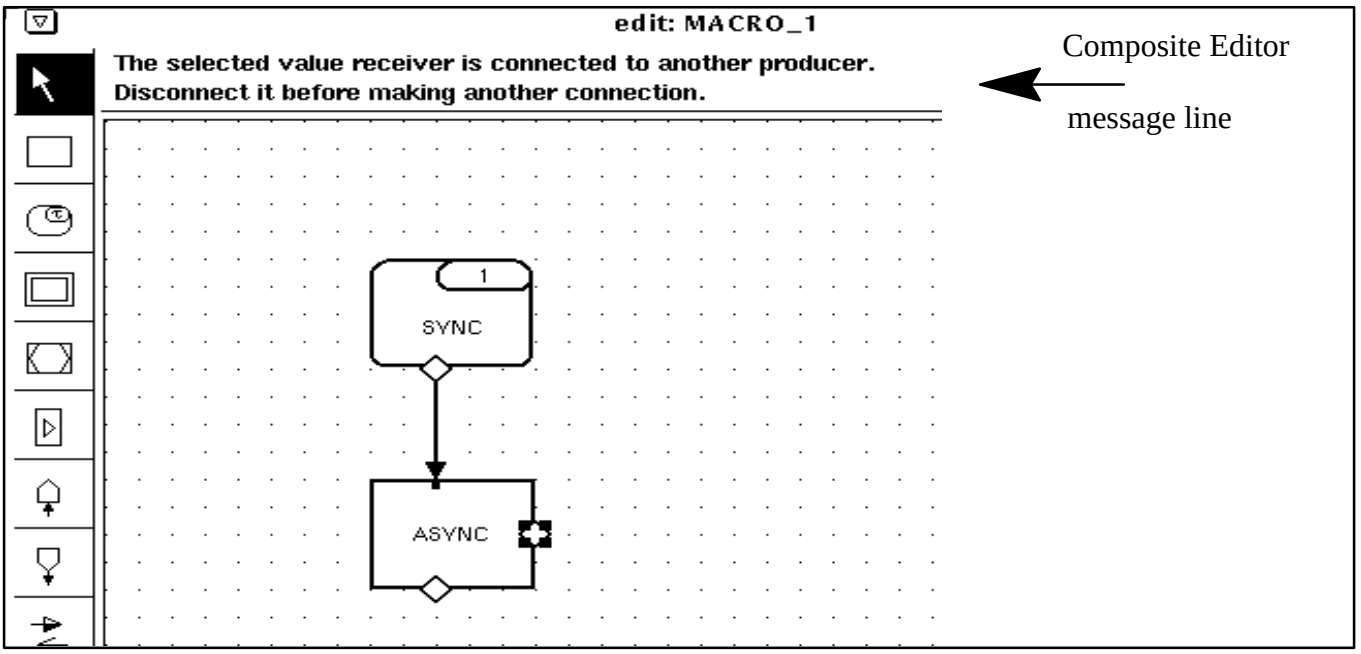


Figure 7 : *Error message displayed in the Composite Editor message line*

Copyright per DIN 34

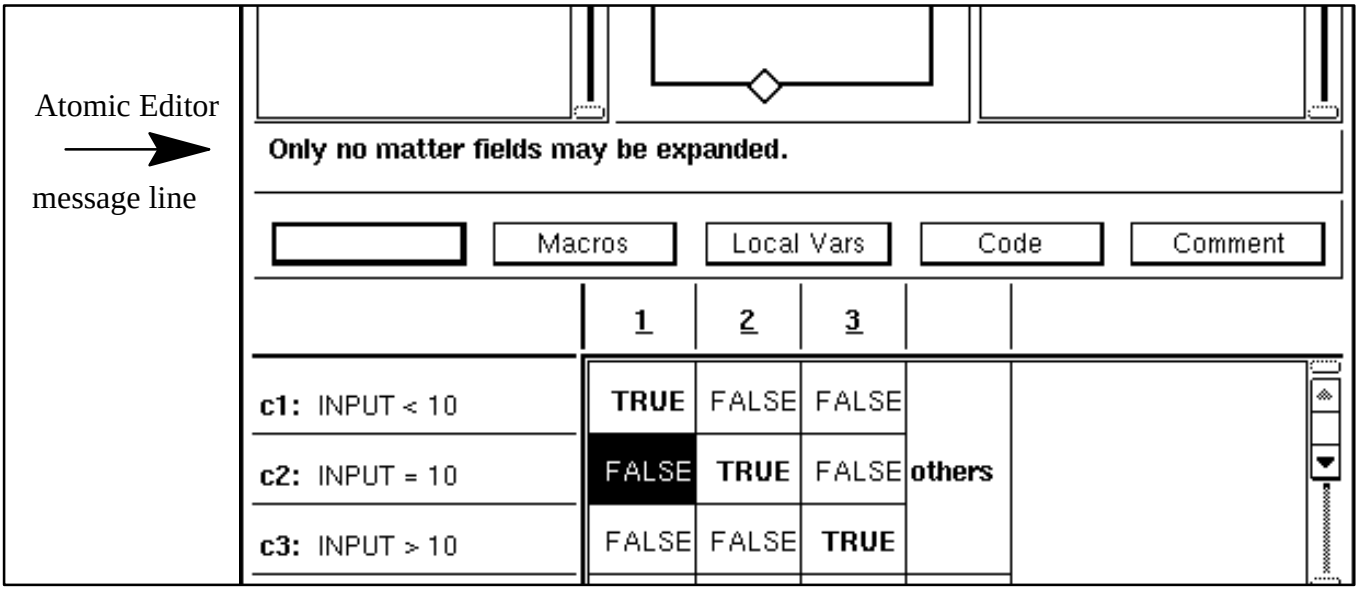


Figure 8 : *Error message displayed in the Atomic Editor message line*

D-2.2.1.3 Error messages during model configuration

Errors and warnings found during the MDE rule check are displayed in an extra scrollable Rule Check error window.
(see Figure 9)

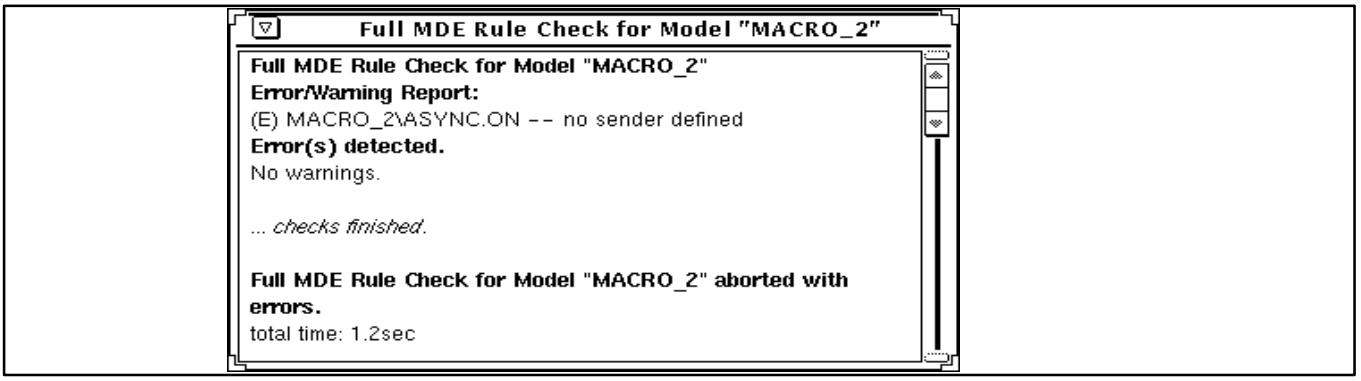

Full MDE Rule Check for Model "MACRO_2"
Full MDE Rule Check for Model "MACRO_2"
Error/Warning Report:
(E) MACRO_2\ASYNC.ON -- no sender defined
Error(s) detected.
No warnings.
... checks finished.
Full MDE Rule Check for Model "MACRO_2" aborted with errors.
total time: 1.2sec

Figure 9 : *The CSS MDE rule check window*

As soon as the Ada compiler starts the transcript window shows the protocol of the proceeding.

The following example gives an impression how to read the transcript window contents.

... compiling model functions ...	*Compilation starts ....*
(E) CTGConnection Protocol Error	*CSS internal error number*
No.: -40 (Ada Error)	
Error Description:	
COMPILATION_TERMINATED_WITH_SYNTAX_ERROR	*CSS internal error description*

Figure 10 : *Transcript window contents – First part*

Copyright per DIN 34

CSS describes the error with an error number and an error description in plain text.

incorrect line 29

ALSYS error message

```
Compiled: "MACRO_4\ASYNC"
    Return code: ERROR: Syntax Error
(E) Error Description:
One error detected. No warnings issued. Compilation failed.
     29              OUTPUT := INPUT ;
                                <-1->
1   **IDE There is a type inconsistency in this expression.

      ======================== More Information ========================
      -> The context requires the following type(s):
        - UNSIGNED_BYTE at line 137 of CSS_TYPES specification, a derived
      type of UNSIGNED_INTEGER8 at line 122 of NUMERIC_TYPES specification
      -> But the expression has the following type(s):
        - UNSIGNED_INTEGER at line 110 of CSS_TYPES specification, a
      derived type of UNSIGNED_INTEGER32 at line 145 of NUMERIC_TYPES
      specification
      ==================================================================

Alsys Ada V5.5.3
Copyright (C) Alsys, 1985, 1991. All rights reserved.
```

Figure 11 : *Transcript window – Second part*

The incorrect line is displayed with its line number.

The error message refers to the indicated portion of the program line and is printed directly below the line containing the error. The ALSYS compiler error message is built up as follows:

- **1** is the error number (there is only one error detected)
- ****** two asterisks which indicate the severity of the error,
- the error code **IDE** which indicates the part of the compiler which issued the diagnostic message
- and the message text (consist of a description of the compiler error)

The section ”More Information” contains suggested actions to resolve the error condition and/or the rule or language rationale upon which the error depends.

Refer to section D-1.2 for more information about the Alsys Ada Compiler error messages.

In many cases the compiler gives a reference to the related chapter in the **Programming Language Ada Reference Manual** (which is abbreviated ”RM” in the compiler message) with some additional explanatory text. (see Figure 12)

Copyright per DIN 34

A:error: RM 3.3: base type of expression must be signed_byte, line 118 in css_types from file /usr/columbus/users11/css_test/v3/css/basic/common_sources/source_current/css_types_.a	*reference to chapter 3.3 of the Ada Language Reference Manual*

Figure 12 : *Transcript window contents – Third part*

The protocol closes with CSS internal error messages and the statement that the configuration was aborted.

CSS internal error numbers	(E) CTGConnection Protocol Error No.: -128 (Internal Error; CTG SERVER: Unhandled Exception) (E) CTGConnection Protocol Error No.: -1 (Broken Connection; Connection to CTG Server broken) **Configuration for Model "MACRO_4" on csf_hp2 aborted with errors.** total time: 2min 7.2sec

Figure 13 : *Transcript window contents – Fourth part*

Errors produced during DBB or MOCS operations appear inside a pop–up menu with **OK** button. After pressing the **OK** buttons the CSS session proceeds normally.

Copyright per DIN 34

D-3 Test Setup and Execution

D-3.1 VICOS Error Messages

In the following section the message types and message groups are shown as used by VICOS. The source where the messages for the groups/types are generated is given to indicate the error origin in general. The general description of 'event' shows the kind of errors/messages generated for the group/type.

Legend: Severity: Severity Level used in the Message Window
ADV = ADVISORY
ORD = ORDINARY
SEV = SEVERE
FAT = FATAL

Group : Identifies logical Group of messsages
Can be used in TEV to select log events

Type: Identifes type of message

MSG/INFO	Information to the user
ERR	Error Message
WRN	Warning Message
EXC	Monitoring Exception (Soft Limits)
ALRM	Monitoring Exception (Hard Limits)

Source: CGS Product / Process delivering the message

Event: Events raising the messages

Copyright per DIN 34

Group	Severity	Type	Source	Event
SYST	ADV	MSG	VCS	VICOS System Info Messages
	ORD	ERR	VCS	VICOS System Error Messages
	ADV	WRN	VCS	System Warnings
TSCV	ADV	MSG	TSCV	TSCV Internal Event
	ORD	ERR	TSCV	TSCV related Error Messages
	ADV	WRN	TSCV	TSCV related Warnings
TEV	ADV	MSG	TEV	TEV Internal Event
	ORD	ERR	TEV	TEV related Error Messages
	ADV	WRN	TEV	TEV related Warnings
HCI	ADV	MSG	HCI	Messages and Internal Events
	ORD	ERR	HCI/TES	HCI related Error Messages
	ADV	WRN	HCI	HCI related Warnings
HLCL	ADV	MSG	HCI: HLCL Interpreter	HLCL Commands
	ORD	ERR	HCI: HLCL Interpreter	Error Messages on User Input
	ADV	WRN	HCI: HLCL Interpreter	Warnings
TES	ADV	MSG	TES	TES Internal Event
	ORD	ERR	TES	TES related Error Messages
	ADV	WRN	TES	TES related warnings
	ADV	INFO	SAS	Messages received by TES from SAS
MON	SEV	EXC	TES: Monitor	Monitor Exceptions (Soft Limits)
	FAT	ALRM	TES: Monitor	Danger Limit Violations (Alarms)
	ADV	WRN	TES: Monitor	Monitor initiated AP's/commands
	ADV	INFO	TES:	Monitoring related error
DACQ	ORD	ERR	TES	Acquisition Failures
	ADV	INFO	TES	Acquisition Events
DGEN	ORD	ERR	TES	Generation Failures
	ADV	INFO	TES	Generation Events

Copyright per DIN 34

Group	Severity	Type	Source	Event
ARCH	ADV	INFO	TES	Events related to archiving
	ORD	ERR	TES	Errors related to archiving
CMD	ADV	MSG	TES:	Commands received from remote nodes / workstations
	ORD	ERR	TES:	Erroneous commands / Errors in command execution
	ADV	WRN	TES:	Warnings given on commands received
DDS	ORD	ERR	TES: DDS Service	Data Delivery related failures
	ADV	INFO	TES: DDS Service	Data Delivery Warnings
UCLI	ORD	ERR	TES: UCL Interpreter	UCL Interpretation Errors
	ADV	INFO	TES: UCL Interpreter	UCL Interpretation Messages
LOG	ADV	MSG	TES: UCL Interpreter	UCL ”LOG” Statement (= User Events)
UEVT	ADV	INFO	TES: UCL Interpreter	UCL ”LOG” Statement (= User Events)
REPL	ADV	INFO	TES	Items as read from archive (Replay Mode)
	ORD	ERR	TES	Errors during Replay Mode
C DB	ORD	ERR	Products calling MDA	CDB(MDB) Access failures
	ADV	WRN	Products calling MDA	Warnings /Events during CDB(MDB) access
	ADV	INFO	TES	Events during CDB(MDB) access
TRDB	ORD	ERR	Other products calling DBS	TRDB Access failures
	ADV	WRN	Other products calling DBS	Warnings /Events during TRDB access
SAS	ORD	ERR	TES	Error in SAS Handling
	ADV	INFO	TES	Advisory Messages in SAS Handling
	ADV	WRN	TES	Warning Messages in SAS Handling
	ADV	<XXX>	SAS	Messages received from SAS via TES_API <XXX> is type specified by SAS: MSG,INFO,WRN
	ORD	ERR	SAS	Messages received from SAS via TES_API
	SEV	EXC	SAS	Soft Monitoring Exceptions received from SAS
	FAT	ALRM	SAS	Hard(Danger)Monitoring Exceptions from SAS
UUT	ADV	INFO	SAS	UUT Messages received via SAS
TST	ADV	MSG	Test Messages	(for debugging)

Copyright per DIN 34

D-3.2 Test Setup

D-3.2.1 System Setup / Shutdown

stop_cgs (25423) begin.”,
Hosts: cgs–test cgs–test1 csf_hp2 cofprime

cgs_startup (24378) end.

cgs_shutdown (23131) begin.”,
Hosts: cgs–test cgs–test1 csf_hp2 cofprime

stop_cgs_node (27161) end. Errors !”,
Exit code 1
Effective user = cgs_1, NO permission, login as cgsadmin
—> The user cgs_1 has tried to shutdown the system. This is only allowed for
—> user cgsadmin

D-3.2.2 Test Configuration Setup and Verification (TSCV)

D-3.2.2.1 TCSV Error Conditions

The TSCV product runs on a CGS workstation, and can execute in two modes: either as a batch program which executes its task solely on basis of data provided as parameters on the command line, or an interactive program. It uses interfaces provided by other CGS products, but does not provide any interface to other products.

Interactive mode

All error messages created in WIMP mode will be displayed to the user in a pop–up dialog window and sent to the Message Window.

When TSCV executes in the interactive mode, all error conditions not handled internally will be escalated to the user, and resolved by him. No foreseen error condition shall cause the program to halt.
Some errors may however not be resolved by the user of TSCV, but must be resolved by system maintenance, database maintenance etc.

If TSCV for some reason has not been able to perform user authorization TSCV will terminate (after prompting the user). The user account must be established – with sufficient authority to run TSCV.

Copyright per DIN 34

Batch mode

When TSCV starts executing in batch mode runs in batch mode, it will communicate a ’result’ as a Unix completion status. It can be read by the program which invoced TSCV to determine the status upon termination. The following exit codes may be produced by TSCV in batch mode:

- Success 99;
- CCU selection invalid for some reason 100;
- invalid mission name 101;
- invalid element configuration 102;
- invalid system tree version 103;
- invalid CCU name 104;
- invalid CCU version 105;
- invalid system tree node name 106;
- can not connect 107;
- invalid test configuration 108;
- EGSE setup failed 109;
- test failed 110;
- could not create test session 111;
- test configuration in use 112

Copyright per DIN 34

D-3.2.2.2 TCSV Error Messages in the Console Window

*** 'Alarm' from TSCV! The program is no more able ***
*** to 'touch' its lock file to indicate that it is ***
*** alive'. Program exception STORAGE_ERROR occured ***
*** in the program unit called 'ALIVE_TASK'. ***
—>
—> kill process and restart TSCV

Main Program abandoned: Exception WIN_ILLEGAL_HANDLE
—> Check DISPLAY variable: Is is set to ":0.0" ?
—> Change to the correct value: setenv DISPLAY <host>:0.0

D-3.2.2.3 TCSV Error Messages to the Message Window

Another TSCV instance is already executing.
In module: TSCV_MAIN.TSCV
Only one instance of TSCV can be active at a time. This instance of TSCV will quit.

Comms error in DBS_COMMS.INIT_LOCAL
CONNECT_TO_VICOS_PRODUCT returns ETIMEDOUT
—> This message appears when TSCV is started and DBS is not running
—> TSCV will then start DBS; so the message can be ignored
—> In other situations:
—> TSCV is not able to connect to DBS;
—> Verify DBS processes are active

Comms error in DBS_COMMS.SEND_WITH_ACKNOWLEDGE_TO_CENTRAL
SEND_MESSAGE returns ETIMEDOUT
Problem with connection to DBS: DBS_COMMUNICATION_PROBLEM Stop and restart TSCV
—> DBS is in a bad shape
—> shutdown dbs (kill processes)
—> restart TSCV (will restart DBS)

Could not convert SID to PATHNAME: Error retrieving MDB data.
—> Was there a previous error indicating access problem to the MDB ?
—> Is there an old testconfiguration used which has no correspondence in the MDB anymore ?

DBS is in Error (pop up window)
—> the overall status of DBS is NOT_OK
—> Is the DISK full ?
—> Remove test sessions from the TRDB
—> cleanup the disk partition where the TRDB is located
—> must have the percentage of the overall space free
—> that is defined in the file $CGS_HOME/etc/cgs_configuration.xml

DBS error:INVALID_SESSION_NAME&&The error occured when creating session xxxxx
—> The session name given when creating a test session is invalid
—> The session name must be in CAPITAL letters !

Copyright per DIN 34

Default test session already in use by other.

Only one test configuration may utilize the default test session at a time.

Deleting session:Failed to delete session <session_name>.:SESSION_IS_USED

—> The session is currently in use (open or evaluated)
—> If no user is active: There was probably a crash of a tool when accessing the session
—> Use DBS recover scripts to force the removal of the session:
—> As cgs_administrator execute $DBS_HOME/util/sun4/delete_sessions_like <session_name>

Error commanding TES: Setup failed, error in TES_01

– COMMUNICATION_ERROR in INIT to TES_01
—> INIT command from TSCV to TES failed
—> could be several reasons:
—> a) Connection to MDB failed
—> there was no CCU written by the ”Generate SCOE files”
—> function within I_MDB
—> b) Programming error in TES
—> Write SPR resp. use correct TES version
—> c) TSS is not running
—> check if the tss_tsp process is running on the test node
—> Connection to DBS failed
—> check if DBS is running; Can other process connect to it ?

Error commanding TES: Setup failed, error in TES_01

– OTHER_ERROR in INIT to TES_01
—> INIT command from TSCV to TES failed
—> could be several reasons:
—> a) Data Loaded from MDB are erroneous
—> Use MDB consistency checker to verify data loaded from MDB
—> (on CCU as used for the current Test Configuration)
—> b) Programming error in TES
—> Write SPR resp. use correct TES version
—> C) an exception occured while initialising HCI, DBS, TSS or the local data pool within TES

Error executing Set–up: Error when singalling new global CCU to HCI

Error sending message:ETIMEDOUT
—> For each HCI which is in the test configuration, but not running
—> such a message occurs
—> Ignore if HCI is really not running

ERROR in L_CY_EVENT_SESS_SEND.CREATE_LOCAL_FILE

ADT_PACKED_FILE.CREATE fails for file /testnode/cgs–test/TSCV_01.11–02–98_11:34:”, E_CREATE_FILE_FAILED. Cause: Cannot create file. Consequence: CREATE_LOCAL_FILE failed. Recovery act: none. Debug info: dbs_local_cyclic_event_sess_sender.a

Error occured when starting TES_01

In module EXTIF_OS.START_APPLICATION Symptom: CONNECTION_TIMEDOUT:
Setup failed: Launching TES_01 Could not start TES_01
—> The Process_Creation_Server process is not running or not
—> answering when TSCV tries to launch a test node (TES_01)

Copyright per DIN 34

—> Get UNIX node name for TES_01 from
—> $CGS_HOME/config/SYSTEM_TOPOLOGY_TABLE
—> Verify all servers are running. To start Process_Creation_Server:
—> Execute ”$CGSI_HOME/bin/common/Start_Process_Server”

External error when calling MDB to set local default

In module MDB_SESSION.SET_LOCAL_DEFAULT Symptom: ORACLE_ERROR.
—> Is Oracle running ? Check on DB server node if Oracle processes are up.
—> Is the MDB installed correctly ?

EXTIF_DBS.CONNECT_TO_DBS: Error when calling SystemTopologyTable.Who_Am_I:

EXTIF_DBS.CONNECT_TO_DBS: Error when calling
SystemTopologyTable.Who_Am_I:TYPE_UNKNOWN
—> Is TSCV entered on this node in the
—> $CGS_HOME/config/SYSTEM_TOPOLOGY_TABLE ?

Failed to access the Mission Database (MDB).

– EXTIF_MDB.GET_CCU_VERSION_LIST:Operation MDB_SESSION.READ_SIDS raised exception”,
The initiated operation can not be completed because of an internal error in the Mission DB provided procedures. Please contact the system manager”

–EXTIF_MDB.GET_CCU_VERSION_LIST:Operation MDB_SESSION.GET_LIST_OF_TC_IDENTIFIERS
The initiated operation can not be completed because of an internal error in the Mission DB provided procedures. Please contact the system manager”);

Failed to check TSP status on node <unix–node>

EXTIF_TSS.VERIFY_LOCAL_TSP_STATUS:Operation TSS_NODE_STATUS returned ETSUNKNOWNF”,
The inquiry of the Time Server Process status resulted in an error indication.
Unable to open NTP status file!
The SMT services on cgs–test may be incorrect.”);

Failed to clear the SMT time.”,

EXTIF_TSS.VERIFY_LOCAL_TSP_STATUS:Operation TSS_NODE_STATUS returned ETSUNKNOWNF”,
Unable to open NTP status file!
The SMT services of all nodes of the test configuration remains set–up as before.

Failed to close test session.

EXTIF_DBS.CLOSE_SESSION:Operation DBS_RPI.CLOSE_EXECUTION_SESSION returned COMMS_TIMEOUT
Communication time out occurred.The test session remains open.

Failed to launch test node TES_xx

EXTIF_OS.START_APPLICATION:Operation OS_BINDING.START_APPLICATION raised excepti
Failed to invoke CGS script.

Copyright per DIN 34

–> Check that the process creation server is running,
–> start test system may solve the problem.The present state of test node TES_xx is unknown.

Failed to open file.

TSCV_TRDB_SELECTOR.OPCS_SHOW_SESSION:Exception raised: USE_ERROR
Opening file /gsaf_home/4.1.1/tscv/data/session_view_file.tmp for read or write failed.

Failed to initialize CGS connections.",

EXTIF_TSS.VERIFY_TSS_INTERFACE:The TSS interface returned
TSP_IF_NOT_INITIALISED",
The CGS services are required to run in order
to provide for CGS inter–application communication.
The TSS interface has not been initialized!
None of the CGS application can be controlled by TSCV.

Failed to log to DBS, not connected!

Event: TSCV is stopping.

Failed to setup test node TES_xx

EXTIF_TES.INIT_TES:Operation TES_RPI.INIT returned COMMUNICATION_ERROR",
Communication problem with the test node.
Try to shut–down and then restart the test node.
The test node TES_xx will not be setup, but will participate in this test configuration.

Failed to shut–down node(s)!

EXTIF_OS.SHUTDOWN_NODE:POSIX_ERROR
Failed to finally shutdown the CGS services on
the nodes: cgs–test1 csf_hp2,
using script stop_cgs_node located at $GSAF_HOME/cgs/bin/common.
Failed to invoke CGS script using remote shell. Verify access rights and correct the environment.

Failed to shut–down the Test Result DB.

EXTIF_DBS.STOP_CENTRAL_DBS:Operation DBS_RPI.STOP_CENTRAL returned
DBS_COMMUNICATION_ERROR
The error indicates that the TRDB has to be started.
Invoking check status for the test configuration may solve the problem.

Failed to start–up nodes

EXTIF_OS.STARTUP_NODE:POSIX_ERROR",
Failed to start the CGS services using script start_cgs_node located at
$GSAF_HOME/cgs/bin/common.
Unless the CGS services are not running correctly, the test node cannot be operated.

Failed to unlock TSCV.

TSCV_LOCK.UNLOCK:Exception raised: NAME_ERROR
The TSCV lock file is probably not deleted
An internal TSCV program error was detected. Please report this error to the system manager.
This will not cause any problems. Even with a lock file present, TSCV will start if it is detected that the lock file is unchanged

Failed to update the TSCV lock file.

T_ALIVE_TASK:NAME_ERROR
The TSCV lock file ensures to start only one TSCV instance at a time. Cyclically updating the lock file located at $GSAF_HOME/tscv/data failed.
Using more that one TSCV instance at a time may result in conflicting test system setups.

Copyright per DIN 34

Failed to verify status of test node TES_xx!

EXTIF_TES.GET_TES_STATUS:Operation TES_RPI.READ_STATUS
returned NO_SERVER_CONNEC
TES_xx is not running any more.
The status will be rendered 'No contact'

Loading test configuration.

In module: TC_SELECTOR.LOAD_TC_TEMPLATE",
The test configuration contains node(s) that is not defined in the System Topology Table.
The missing node(s): HCI_03

No CCU is set in the Mission Data Base.

The status of the active test configuration no. <n> was verified.
Further set–up of TES instances will not be possible for this
test configuration unless stopped and set–up again.

Problem with connection to DBS: CONNECT_PROBLEM Stop and restart TSCV

Unexpected error occured. in module EXTIF_MDB.CLOSE_CONNECTION Symptom:
USE_ERROR. Please issue SPR!
—> In TSCV engineering version: Wrong sequence in DBS startup
—> Restart TSCV

Setup with node not in System Topology.",
In module: L_TCS_CONTROLLER.OPCS_SETUP",
Node HCI_xx is participating in the test configuration
to be setup, it is not, however, found in the System Topology table.

SMT error (1300) on host gsrf_dbs. Already member of a SMT domain

—> Maybe the TSP process on the node was already initialised in a
—> previous setup —> ignore as long as the previous setup was the same

SMT error (1302) on host gsrf_dbs. SMT–client(s) not reachable

—> Maybe the TSP process on the node is not runningwas already initialised in a
—> previous setup —> ignore as long as the previous setup was the same

SMT service is not setup on <node>!",
In module: MAIN.CHECK_TSP_STATUS",
All SMT based operation will not function on this node.
Checking status of the test configuration may solve the problem.
You may either continue checking the TSP status on remaining nodes, continue but suppress warnings or cancel further TSP checking.

Status of TRDB has changed.

The test session MARTIN associated to test configuration no. <n> seems to be closed since last time TSCV ran.
All related events will be logged to the default test session
You may create a new test session for this configuration

The test configuration is active!

In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TES",
Nodes hosting the DBS, the local TSCV or in use by
other active test configurations are excluded.");

Status check failed.

EXTIF_DBS.GET_DBS_STATUS:Operation DBS_RPI.GET_TRDB_INFO

Copyright per DIN 34

returned COMMS_TIME_OUT",TRDB availability was to be checked.
A problem was encountered when contacting DBS. Communication time out occurred.
Check network and communication services.

Test configuration(s) are active!

In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TEST_SYSTEM
Confirm shut–down of the test system.

Test configuration has been modified!

All modifications will be lost when unloading the test configuration.

Test configuration no <n> is undefined

– REPOSITORY_TC.RETRIEVE_OLD_NODE_INIT_SYNCH_STATUS_:NAME_ERROR",
The description of the idle test configuration is not available in the Mission DB.
Program exception was raised when accessing save files for reading data.
The test configuration can not be used for further tests,but can still be viewed.
Please unload it

– L_TCS_CONTROLLER.OPCS_SETUP:Setup Denied",
The description of the test configuration is not available in the Mission DB.
An internal TSCV program error was detected.
Please report this error to the system manager. Setup can not be performed

Test node TES_xx rejected the stop request!

In module: L_TCS_TES_COMMANDING.OPCS_STOP_TEST

Test session <name> is already used.

L_TCS_SESSION_MANAGEMENT_SELECTOR.OPCS_APPLY:Session <name> already used.
No test session is created.

The test configuration is active!

In module: L_TCS_CONTROLLER.OPCS_SHUTDOWN_TES
Nodes hosting the DBS, the local TSCV or in use by other active test configurations are excluded.

Unable to read status for TES_01.

Unexpected status of TES_xx!

The test nodes were interrogated for current status.
Node TES_xx indicates that the status has changed from when TSCV was last shut–down.

Unexpected program error (check icon file).

in module GUI_CONTROLLER.CREATE_TSCV_ICON Symptom: WIN_ERROR. Please issue SPR!

—> installation problem (icon file not found)
—> can be ignored during integration of CGS

Copyright per DIN 34

D-3.2.2.4 TCSV Error Messages in Pop Up Window

Internal program error

This message can appear, if operating system resources like disk space, swap space, etc. are not available. Call the system operator to check it.

ERROR!
Internal program error.
An internal TSCV program error was detected.
Please report this error to the system manager.
This TSCV instance may not operate orderly any more.
You may quit TSCV , and restart it.
OK

Figure 14 : *The Internal Error window*

Copyright per DIN 34

D-3.3 Test Execution

D-3.3.1 Messages from HCI (Workstation)

D-3.3.1.1 Messages on Console Window

**** MAIN PROGRAM ABANDONED — EXCEPTION ”constraint_error” RAISED**

—> if the workstation your are running your X–Server on is in a
—> bad state, this exception could occur
—> Re–Initialise your openlook env. or re–boot your workstation
—> Are the resources required exceeding the resource available ?
(e.g. is Text.MaxDocumentSize: 400000 set ?)

**** MAIN PROGRAM ABANDONED — EXCEPTION ”CONF_CODE_NOT_FOUND” RAISED**

—> The TSS configuration file is not present or contains
—> wrong data
—> Check in $TSS_HOME/config/tss_conf_file
—> should contain (V4.1):

```
NTP_STATUS_FILE /usr/tmp/ntp_status
SHARED_MEM_KEY_FILE /config/tss_shmkeyfile
SEMAPHORE_KEY_FILE /config/tss_semkeyfile
SMT_MONITOR_DELAY 0.1
```

Copyright per DIN 34

D-3.3.1.2 Messages in Message Window

Alarm was not acknowledged!
Data Base Server Node: Alarm was not acknowledged!
—> In the system advisory window the status of the DB Server is
—> not ok. In the window the user did not acknowledge (i.e.
—> check box was not activated).
—> Ignore or activate

Alarm was not acknowledged!
<Sunsystem Name of Node>: Alarm was not acknowledged!
—> In the system advisory window the status of the Test Node where the subsystem is defined for
—> is not ok (i.e. there was at least one value out of limit)
—> In the window the user did not acknowledge (i.e.
—> check box was not activated).
—> Ignore or activate

Alarm acknowledged!
Data Base Server Node: Alarm acknowledged!
—> The operator has acknowledged the NOT_OK status in the system
—> advisory by activating the checkbox

Can't load synoptic display!",
Synoptic display:

CCU consistency not guaranteed.
The loaded CCU is not in a valid status. Please use the Consistency Checker for the CCU.
—> When connecting to the CCU, HCI gets the status reported from
—> the MDB. Means, that the consistency checker has not yet been run.
—> Can be ignored, if the status of the data is known. If not, the consistency
—> checker should be run.

CCU consistency status is local invalid
—> When connecting to the CCU, HCI gets the status reported from
—> the MDB. Means, that the consistency checker has found errors.
—> Can be ignored, if the status of the data is known and the errors reported
—> by the consistency checker can be ignored.

Checkout of "DVtools" failed
DVtools: cannot find license file (No such file or directory)
Product "DV–Tools" not validated — Error code 909
—> There is no license for your environment for the DATAVIEWS tool
—> Check with your system administration
—> Check if the License Server of DataViews is running

Command Facility: Illegal default path!
Ignored illegal default path <path>!
—> problem during screen setup when loading the command window:
—> the given default path was not a valid path in the selected CCU
—> Path must be virtual,CDU,system tree node
—> (i.e. must have attribute 'path_select'
—> B2 problem (17.1.96): CDU is not accepted in screen setup

Copyright per DIN 34

Comms error in DBS_COMMS.INIT_LOCAL

SETUP_COMMUNICATION_CONFIGURATION returns NW_BADBINDPORT

—> when starting up HCI: could mean that another HCI is

—> already active on the same node !

Connected to node (TES_xx).,

Connected to <pathname> (TES_xx).");

Connecting to configuration control u...",

Element configuration is <element>, mission is <MISSION>, system tree version is <version>, CCU version is <pathname> <CCU Name> <Version>");

Could not connect to test node \EUREC...

Could not connect to test node \EURECA\EGSE\PAULS_TEST\TEST_NODE (TES_01)!

—> the indicated test node was in the test configuration

—> but was not in the correct mode when HCI tried to connect to it

Could not load message definitions!

Error status was DEFINITIONS_NOT_FOUND!

—> During initialisation HCI loads data from file

—> $CGS_HOME/config/errmsg_defs.dat

—> This file has not been found

Could not load test configuration!

–Internal error!

—> HCI did not find a correct Test Configuration in file

—> $CGS_HOME/local/config/cgs_test_configuration<n>.dat

–This Online Test Control (HCI) instance is not participating!

Could not request data!

No test node available for that data!

—> request for an enditem has been done

—> The enditem is not provided/monitored by any test node

—> Is the enditem loaded to a test node ?

—> Check Test Configuration and Generated Load_Scoe files

—> Is HCI connnected tothe test node which provides the data ?

—> was there an error during the connection ?

Could not start Online Test Control!

This Online Test Control (HCI_xx) is not participating in an executing test configuration!

—> There is no test configuration active, where the HCI_xx is participating

—> Start Test Configuration via TSCV first !

Disconnected from node (TES_xx).",

Disconnected from test node ...\\TEST_NODE_02 (TES_02).");

Exited Online Test Control.

Online Test Control started

Started by user <user>

—> HCI has been started successfully

Copyright per DIN 34

D-3.3.2 Messages from TES (Test Node)

In the message window:

Note: In the following, the type and group of each message is specified. When sent to the Message Window, the Type is mapped to the Message Window's classification scheme as follows:

ALRM	–> FATAL
EXC	–> SEVERE
ERR	–> ORDINARY
any other	–> ADVISORY

TYPE GROUP Text

ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Raw val: <VALUE> Eng val: <VALUE> Measurement: <PATHNAME>”

ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE> SW Variable: <PATHNAME>”

ALRM MON ”Danger high limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE> Derived value: <PATHNAME>”

ALRM MON ”Danger low limit monitoring exception”
”Danger low limit monitoring exception at <TIME> Raw val: <VALUE> Eng val: <VALUE> Measurement: <PATHNAME>”

ALRM MON ”Danger low limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE> SW Variable: <PATHNAME>”

ALRM MON ”Danger low limit monitoring exception”
”Danger high limit monitoring exception at <TIME> Eng val: <VALUE> Derived value: <PATHNAME>”

ALRM MON ”Danger delta limit monitoring exception”
”Danger delta limit monitoring exception at <TIME> new / old Eng val: VALUE> / <VALUE> <Measurement/SW Variable/Derived value>: <PATH NAME>”

ALRM TES ”<SAS_NAME> disconnected by TES”
”The application <SAS_NAME> got automatically disconnected, because it did not check for commands sent to it within time. The disconnect timeout can be configured by altering parameter TES.API.CONTROLLER.TIMEOUT_PERIOD_FOR_READ_CMD in the CGS configuration file.”

ALRM TES ”Error creating the local SID table”,

Copyright per DIN 34

”Not enough memory could be found to create the local SID table for data distribution”
ALRM TES ”Error creating the remote SID table”,
”The size of the remote SID table is not large enough. Please stop the test system, increase the parameter TES.KERNEL.DISTRIBUTION_TABLE.MAX_NB_SID in the CGS configuration file and restart the system”

TYPE GROUP Text

EXC MON ”<CODED_ERROR_TEXT1>”
”<CODED_ERROR_TEXT2>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Raw val: <VALUE> Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Eng val: <VALUE> SW Variable: <PATHNAME>”
EXC MON ”Nominal high limit monitoring exception”
”Nominal high limit monitoring exception at <TIME> Eng val: <VALUE> Derived value: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Raw val: <VALUE> Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Eng val: <VALUE> SW Variable: <PATHNAME>”
EXC MON ”Nominal low limit monitoring exception”
”Nominal low limit monitoring exception at <TIME> Eng val: <VALUE> Derived value: <PATHNAME>”
EXC MON ”Nominal delta limit monitoring exception”
”Nominal delta limit monitoring exception at <TIME> new / old Eng val: VALUE> / <VALUE> <Measurement/SW Variable/Derived value>: <PATH-NAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Raw val: <VALUE> Eng val: <VALUE> Measurement: <PATHNAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Eng val: <VALUE> SW Variable: <PATHNAME>”
EXC MON ”Expected value monitoring exception”
”Expected value monitoring exception at <TIME> Eng val: <VALUE> Derived value: <PATHNAME>”

TYPE GROUP Text

Copyright per DIN 34

ERR ARCH ”Automatic enable archiving.”
”Automatic enable archiving after system error. Archive queue is broken. Complete evaluation
or replay of archive session is impossible.”
ERR ARCH ”Could not store archive file.”
”Failure calling DBS.Store_Raw_Data_File.”
ERR ARCH ”Could not create archive file.”
”Could not create archive file. Archiving disabled.”
ERR ARCH ”Could not close archive file.”
”The archive file <FILENAME> could not be closed.”
ERR ARCH ”Could not close archive file.”
”The archive file <FILENAME> could not be closed. Archiving disabled.”
ERR ARCH ”Could not create archive file.”
”Unexpected error while creating automatically a new archive file. Archiving disabled.”
ERR ARCH ”Illegal archive cycle.”
””
ERR ARCH ”Could not archive ADU.”
”Could not archive ADU. Archiving is disabled.”
ERR ARCH ”Could not archive ADU–request.”
”Could not archive ADU–request. Archiving is disabled.”
ERR ARCH ”Could not archive GDU.”
”Could not archive GDU. Archiving is disabled.”
ERR ARCH ”Could not archive SMT update.”
”Could not archive SMT update. Archiving is disabled.”

WRN ARCH ”Warning: Disk–space below limit.”
”available disk–space is: <NNNN> bytes.”
WRN ARCH ”Warning: Could not read free disk space.”
”TES internal error : Failure reading free disk space from System_Server: <NNNN>”

TYPE GROUP Text

ERR CDB ”Error in Loading Database”
”There was an error when unpacking measurement data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking gdu data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking adu data from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking monitor lists from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking gdu lists from file”
ERR CDB ”Error in Loading Database”

Copyright per DIN 34

”There was an error when unpacking simulated values file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking Automated Procedures from File”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking node defs from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking sas names from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking user msg from file”
ERR CDB ”Error in Loading Database”
”There was an error when unpacking user libraries from file”
ERR CDB ”Error in Loading Database”
”$MDA_HOME/data is not a directory”
ERR CDB ”Error in Loading Database”
”$MDA_HOME/data does not exist”
ERR CDB ”Error in Loading Database”
”A directory could not be created in the MDA data directory”
ERR CDB ”Error in Loading Database”
”A directory was missing in the MDA data directory”
ERR CDB ”Error in Loading Database”
”A file from the MDA data directory could not be opened”
ERR CDB ”Error in Loading Database”
”Error when closing file”
ERR CDB ”Error in Loading Database”
”Error when reading file”
ERR CDB ”Error in Loading Database”
”No node name in load point string”
ERR CDB ”Error in Loading Database”
”Load point pathname is missing”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because alternative is UNDEFINED”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because acquisition alternative is UNDEFINED”
ERR CDB ”Error in configuration data.”
”<PATHNAME> has an invalid calibration definition and therefore will never get a valid value.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the engineering value is of an unknown type.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the raw value is of an unknown type.”
ERR CDB ”Error in configuration data.”
”<PATHNAME> has an inconsistent default value type.”
ERR CDB ”Error in configuration data.”

Copyright per DIN 34

”<PATHNAME> is discarded because the HK value with ID: <NNNN>
is not supported. Please check in the MDB”

ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the HK value with ID: <NNNN>
has already been linked by <PATHNAME>”

ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because there is a type mismatch between
the HK value with ID: <NNNN> and the software variable.”

ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because the enditem type is not a measurement,
software variable or derived value”

ERR CDB ”Error in configuration data.”
”Monitoring will not be performed on <PATHNAME> because no limit set is defined”

ERR CDB ”Error in configuration data.”
”Monitoring will not be performed on <PATHNAME> because limit set <NNNN> is invalid.”

ERR CDB ”Error in configuration data.”
”The enditem <PATHNAME> in the monitor list <PATHNAME> is not known.”

ERR CDB ”Error in configuration data.”
”<PATHNAME> will not be processed from ADU <PATHNAME> because it is not defined
in the test node.”

ERR CDB ”Error in configuration data.”
”<PATHNAME> is discarded because it is not acquired by an ADU.”

ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is invalid because it has a cyclic dependency with

<PATHNAME>”

ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous because it uses <PATHNAME> that is erroneous”

ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous because it uses <PATHNAME> that is not existing
in the test node”

ERR CDB ”erroneous derived value”
”derived value <PATHNAME> is erroneous. An error occurred during initialisation”

ERR CDB ”error in dependency table”
”derived value <PATHNAME> linked to the software variable <PATHNAME> is not existing”

ERR CDB ”error in dependency table”
”<PATHNAME> is identified as a software variable in the dependency table but it has not been

Copyright per DIN 34

loaded”
ERR CDB ”error in dependency table”
”derived value <PATHNAME> linked to the adu <PATHNAME> is not existing”
ERR CDB ”error in dependency table”
”<PATHNAME> is identified as an ADU in the dependency table but it has not been loaded”
ERR CDB ”Error in configuration data.”
”The TC Verification definition of <PATHNAME> is incorrect and will be discarded :
item at position <NNNN> is unknown on the test node.”
ERR CDB ”Error in configuration data.”
”The enditem <PATHNAME> in the GDU list <PATHNAME> is not known.”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking swop command data from file”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking application ID data from file”
ERR CDB ”Error in Loading SW command Database”
”There was an error when unpacking response packet data from file”
ERR CDB ”Error in Loading SW command Database”
”$MDA_HOME/data is not a directory”
ERR CDB ”Error in Loading SW command Database”
”$MDA_HOME/data does not exist”
ERR CDB ”Error in Loading SW command Database”
”A directory could not be created in the MDA data directory”
ERR CDB ”Error in Loading SW command Database”
”A directory was missing in the MDA data directory”
ERR CDB ”Error in Loading SW command Database”
”A file from the MDA data directory could not be opened”
ERR CDB ”Error in Loading SW command Database”
”Error when closing file”
ERR CDB ”Error in Loading SW command Database”
”Error when reading file”
ERR CDB ”Error in Loading SW command Database”
”No node name in load point string”
ERR CDB ”Error in Loading SW command Database”
”Load point pathname is missing”
ERR CDB ”Error in Loading Database”
”$MPS_HOME is no directory”
ERR CDB ”Error in Loading Database”
”$MPS_HOME not defined”
ERR CDB ”Error in Loading Database”
”MDB directory structure is incomplete”
ERR CDB ”Error in Loading Database”

Copyright per DIN 34

”Directory could not be created on $MDB_HOME”
ERR CDB ”Error in Loading Database”
”Error within opening the Load SCOE file. Verify that Load SCOE files have been created”
ERR CDB ”Error in Loading Database”
”Error within closing the Load SCOE file”
ERR CDB ”Error in Loading Database”
”Read from Load SCOE file error”
ERR CDB ”Error in Loading Database”
”Error within Load SCOE file lock/unlock”
ERR CDB ”Error in Loading Database”
”No node name in load point string”
ERR CDB ”Error in Loading Database”
”Load point pathname is missing”
ERR CDB ”Error in Loading Database”
”Probably a memory problem.”

TYPE GROUP Text

ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition, but it cannot be loaded from the MDB”
ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition, but it has parameters”
ERR COND ”Could not start AP.”
”AP <PATHNAME> should have been started as a result of a condition, but it is already running”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : IN_RANGE conditions cannot be defined interactively.”
ERR COND ”Erroneous condition”
”<PATHNAME> is not a scalar type and cannot have a IN_RANGE condition check”
ERR COND ”Erroneous condition”
”<PATHNAME> references itself for selecting a limit set on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> has no monitoring limits.”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : The limit set <NNNN> is undefined for enditem
<PATHNAME>”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : The limit set number <NNNN> is illegal”
ERR COND ”Erroneous condition”

Copyright per DIN 34

”<PATHNAME> references itself for enabling / disabling the processing on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> is a SW variable linked to an HK
data and its processing cannot be enabed / disabed on condition”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : <PATHNAME> is unknown on the test-node”
ERR COND ”Erroneous condition”
”Erroneous condition for <PATHNAME> : ITEM is unknown on the testnode”
ERR COND ”Erroneous condition”
”CONDITION_ITEM is unknown on the testnode”

INFO COND ”Condition triggerred.”
”Enabling the processing of <PATHNAME>”
INFO COND ”Condition triggerred.”
”Disabling the processing of <PATHNAME>”
INFO COND ”Condition triggerred.”
”Setting limit set <NNNN> for <PATHNAME>”
INFO COND ”Condition triggerred.”
”Starting AP <PATHNAME>”
INFO COND ”Enabling conditions.”
”The conditions defined by <PATHNAME> are enabled”
INFO COND ”Disabling conditions.”
”The conditions defined by <PATHNAME> are disabled”

TYPE GROUP Text

ERR DACQ ”Could not start simulation on adu”
”The raw value (sid) <SID_NUMBER> (type: <RAW_VALUE_TYPE>) is not of type INT,
U_INT or FLOAT_EGSE”
ERR DACQ ”Wrong ADU definition in MDB.”
”The global length of the ADU in MDB is wrong. Please check the size of <PATH-NAME>
and its simulated counterpart in MDB.”
ERR DACQ ”Could not set raw value in simulated adu”
”The raw value (sid) <SID_NUMBER> (type: <RAW_VALUE_TYPE>) is not of type INT,
U_INT, FLOAT_EGSE or BYTE_STREAM”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. ADU is undefined.”
ERR DACQ ”Could not construct ADU.”

Copyright per DIN 34

”Could not construct ADU. ADU type mismatch.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. Illegal SID in ADU description.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. ADU value not defined.”
ERR DACQ ”Could not construct ADU.”
”Could not construct ADU. Index out of range.”
ERR DACQ ”Nothing to enable for acquisition”
”Pathname <PATHNAME>”
ERR DACQ ”Could not request ADU”
”Could not request ADU <PATHNAME> from <SAS_NAME>”
ERR DACQ ”Could not disable ADU request”
”Could not disable ADU <PATHNAME> from <SAS_NAME>”
ERR DACQ ”Parameter of ENABLE_MONITORING ignored”
”Pathname <PATHNAME> is already acquired. Parameter ADU is ignored”
ERR DACQ ”SUPPLY_ADU not allowed in current mode”
”TES must be started before SUPPLY_ADU is called”
ERR DACQ ”Update of software variable failed.”
”<PATHNAME> is not a software variable. Request from <NAME> refused.”
ERR DACQ ”Update of software variable failed.”
”<PATHNAME> is a housekeeping value. Request from <NAME> refused.”
ERR DACQ ”Write Software value failed.”
”Type mismatch. <PATHNAME> is of type <TYPE>, but the type <TYPE> was tried inserted.
Request from <NAME> refused.”
ERR DACQ ”Unsupported SINGLE_BIT type”
”The raw value at index <NNNN> of ADU <PATHNAME> is SINGLE_BIT, which is not
supported. Use U_INT.”
ERR DACQ ”Unknown raw value alternative”
”Raw value alternative <TYPE> not suitable for <structured ADU/unstructured ADU/CCSDS
packet> <PATHNAME>.”
ERR DACQ ”Invalid length field in CCSDS packet”
”Length of CCSDS_PACKET is invalid. Please check the length field in the primary header for
<PATHNAME> in the MDB.”
ERR DACQ ”Unknown ADU type”
”Procedure GET_RAW_VALUE”
ERR DACQ ”Error when processing ADU”
”ADU: <PATHNAME>; measurement no. <NNNN>: exception <EXCEP-TION_NAME>”
ERR DACQ ”Internal Error”

Copyright per DIN 34

”Internal error: Received empty ADU.”

ERR DACQ ”Internal Error”
”Set new CCSDS APID returns unexpected error for <PATHNAME>. ”

ERR DACQ ”Incorrect checksum in ADU”
”The ADU <PATHNAME> has an incorrect checksum.”

ERR DACQ ”Data interruption”
”The ADU <PATHNAME> has been interrupted. Processing ADU with sequence number
<NNNN>”

ERR DACQ ”Data suspended”
”The ADU <PATHNAME> is suspended.”

ERR DACQ ”Attempt to read value failed”
”<PATHNAME> is not a measurement, sw variable or derived value”

ERR DACQ ”Attempt to read value failed”
”Value of <PATHNAME> is invalid (not acquired or incorrect due to previous error.)”

ERR DACQ ”Attempt to read raw value failed”
”<PATHNAME> is not a measurement.”

ERR DACQ ”Attempt to read raw value failed”
”The raw value of <PATHNAME> has not yet been acquired.”

ERR DACQ ”Failed to get monitoring status.”
”<PATHNAME> is not a measurement, sw variable or derived value”

ERR DACQ ”Stop acquisition rejected”
”<PATHNAME> is monitored and can therefore not be disabled for acquisition.”

ERR DACQ ”Failed to turn EVL <ON/OFF>”
”The engineering value logging cannot be changed. <PATHNAME> is unknown.”

ERR DACQ ”ADU service announced again”
”The application <SAS_NAME> has already announced its ADU service.”

ERR DACQ ”Withdraw ADU service failed.”
”Withdraw ADU service failed because the application <SAS_NAME> as not announced
its ADU service.

ERR DACQ ”Cannot set SAS reference for ADU”
”The ADU <PATHNAME> is currently acquired. Its SAS reference cannot be changed”

ERR DACQ ”Cannot set SAS reference for ADU”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”

ERR DACQ ”Data conversion failure”
”String too long (pathname =<PATHNAME>)”

ERR DACQ ”Data conversion failure”
”String too long (hk_id =<NNNN>)”

ERR DACQ ”Internal Error”
”Internal error: ADU with ADU_ID <NNNN> not in local database”

Copyright per DIN 34

ERR DACQ ”Could not read TRDB INFO HK–values”
”DBS TRDB INFO HK–values will not be written. Received return status <STATUS>
from DBS_RPI_FOR_TEST_NODE.GET_TRDB_INFO.”
ERR DACQ ”Could not read disk HK–values”
”DBS disk HK–values will not be written. Received return status <STATUS>
from DBS_RPI_FOR_TEST_NODE.GET_DISK_PARAMETERS.”
ERR DACQ ”Could not read Printer HK–values”
”DBS Printer HK–values will not be written. Received return status <STATUS>
from DBS_RPI_FOR_TEST_NODE.CHECK_PRINT.”
ERR DACQ ”Could not read oracle HK–values”
”DBS oracle statistics HK–values will not be written. Received return status <STATUS>
from DBS_RPI_FOR_TEST_NODE.GET_ORACLE_STATISTICS.”
ERR DACQ ”EGSE node not found”
”EGSE node with pathname <PATHNAME> not found in the internal TES database”
ERR DACQ ”EGSE sw not found in internal database”
”The EGSE sw: <NNNN>”
ERR DACQ ”Internal Error”
”Internal error: <PATHNAME> is not an ADU.”
ERR DACQ ”Internal Error”
”Internal error: ADU <PATHNAME> not in local database.”
ERR DACQ ”Read Value Error”
”The supplier <SAS_NAME> of the ADU <PATHNAME> has not announced its ADU
service.”
ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Measurement undefined (SID: <NNNN>). Pathname:
<PATHNAME>”
ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Simulator is stopped.”
ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Measurement unknown (SID: <NNNN>). Pathname:
<PATHNAME>”
ERR DACQ ”Could not set value in ADU.”
”Could not set value in ADU. Illegal SID.”
ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Constraint error: Bit location out–of–range.”
ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Number of modifying bits must be greater than
0 and less or equal to <NNNN>.”
ERR DACQ ”Could not set bits in ADU.”
”Simulator is stopped.”
ERR DACQ ”Could not set bits in ADU.”

Copyright per DIN 34

”Could not set bits in ADU. ADU not found.”

ERR DACQ ”Could not set bits in ADU.”
”Could not set bits in ADU. Illegal SID.”

ERR DACQ ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”ADU not simulated.”
”ADU <PATHNAME> is not being simulated.”

ERR DACQ ”Could not stop simulation ADU.”
”Could not stop simulation ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not stop simulation ADU.”
”Could not stop simulation ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU. ADU <PATHNAME> is not being simulated.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU <PATHNAME>. Simulation is suspended.”

ERR DACQ ”Could not send simulated ADU.”
”Could not send simulated ADU <PATHNAME>. Simulator is stopped.”

ERR DACQ ”ADU request failed”
”The sending of an disable ADU request for the response packet <PATHNAME> to <SAS_NAME> failed with error <STRING>”

ERR DACQ ”Software command timeout”
”Response <PATHNAME> has not been received in time for software command <PATHNAME>”

ERR DACQ ”Incorrect checksum in response packet”
”The response packet <PATHNAME> has an incorrect checksum.”

ERR DACQ ”Wrong transaction_id in response packet”
”Response packet <PATHNAME> received with transaction id=<HHHH>, expected=<HHHH> (the hexadecimal repr. of the transaction_id extracted from the return packet resp. the CCSDS primary header of the telecommand”

ERR DACQ ”Invalid Response Packet”
”Response packet <PATHNAME> contained an invalid CCSDS packet or invalid return parameters”

WRN DACQ ”Data packet overflow – packet discarded”
”Discarding data packet <PATHNAME>. Starting with sequence number <NNNN>. The subsequent data packet of that type will be treated as interrupted”

WRN DACQ ”Write Value Error”
”The application <NAME> supplied the ADU <PATHNAME>, but the acquisition of this ADU is not enabled.”

WRN DACQ ”Internal Error”

Copyright per DIN 34

”SUPPLY_ADU returned with unexpected return status <NNNN>”

INFO DACQ ”ADU Service Announced”
”The application <SAS_NAME> has announced its ADU Service.”
—>The application (SAS) was initialised and announced its ADU service. It is ready now
—> to accept request for acquisition (i.e. enditems associated with this SAS may
—> acquired via START_ACQUISITION command**)**

INFO DACQ ”ADU Service Withdrawn”
”The application <SAS_NAME> has withdrawn its ADU Service.”
—>The SAS has withdrawn its ADU service, i.e. it will not accept any request for
—> enditem acquisition via the START_ACQUISITION command anymore
—> Most SAS will withdraw this service when switched to the RESET state or when
—> re–initialised or when being unloaded.

INFO DACQ ”New SAS reference for ADU”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DACQ ”ROUTE_TO_SAS ignored”
”UCL system library routine ROUTE_TO_SAS ignored in replay mode.”
INFO DACQ ”New SAS reference for ADU’s”
”SAS reference has been set to <SAS_NAME> for all ADU’s under <PATHNAME> that are currently not acquired.”
INFO DACQ ”New SAS reference for ADU’s”
”SAS reference has been set to <SAS_NAME> for all ADU’s under <PATHNAME> that are currently not acquired and where the SAS reference was <SAS_NAME>.”
MSG DACQ ”Set new CCSDS APID”
”For <PATHNAME> set CCSDS APID to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems in <PATHNAME> set CCSDS APID to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems in <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems under <PATHNAME> set CCSDS APID to <VALUE>.”
MSG DACQ ”Set new CCSDS APID”
”For enditems under <PATHNAME> set CCSDS APID from <VALUE> to <VALUE>.”

Copyright per DIN 34

TYPE GROUP Text

ERR DDS ”Dispatch of ADU failed”
”<PATHNAME>: disabling data dispatch due to HCI connection problem”
ERR DDS ”Dispatch of ADU failed”
”<PATHNAME>; dispatch return status: <NNNN>”
=> <dispatch_status>: status when dispatching ADU to HCI
ERR DDS ”Invalid item in data delivery request.”
”<PATHNAME> is not a measurement, sw variable or derived value. The item will be
ignored.”
ERR DDS ”Invalid item in data delivery request.”
”The data requested with SID <NNNN> (pathname <PATHNAME>) is unknown. The item
will be ignored.”
ERR DDS ”Invalid item in data delivery request.”
”The data requested is unknown. The item will be ignored.”
ERR DDS ”Error in Request for dispatch of HK data”
”Hk value <NNNN> does not exist.”
ERR DDS ”Request for ADU dispatch contains error.”
”<PATHNAME> is not an ADU enditem.”
ERR DDS ”Request for ADU dispatch contains error.”
”The data requested is unknown.”
ERR DDS ”Enditem is of wrong type.”
”<PATHNAME> is not an ADU enditem.”
ERR DDS ”Error dispatching data to HCI”
”<STATUS> returned by HCI when delivering data for request ID = <NNNN> of <HCI_NAME>”
=> if <status>: PROTOCOL_ERROR: one possible reason:
—> HCI was shutdown non–nominally when dispatch was
—> running
—> restart HCI
—> re–initialise TES (if errors are still displayed after
—> HCI restart)
ERR DDS ”Invalid request ID”
”Attempted to reuse an existing request ID (<NNNN>)”
ERR DDS ”Invalid data delivery request”
”Invalid data delivery request from <HCI_NAME>: Delivery type undefined.”
ERR DDS ”Repeated confirm”
”Request <NNNN> already confirmed”
ERR DDS ”DDS Stopped”

Copyright per DIN 34

”Attempted to confirm delivery in idle mode”

ERR DDS ”Unknown request ID”
”Request ID <NNNN> is not a registered request”

ERR DDS ”Interruption of data dispatch to HCI”
”An overflow occurred when delivering DDS packet to <HCI_NAME>. A data interruption will occur.”

WRN DDS ”Internal consistency error”
”Missing request for <ADU/enditem/hk–item> <NNNN>”

WRN DDS ”Internal consistency error”
”Missing request list for <ADU/enditem/hk–item> <NNNN>”

WRN DDS ”Internal consistency error”
”Request ID <NNNN> not registered”

Copyright per DIN 34

TYPE GROUP Text

ERR DGEN ”Communication error on Issue TC/stimuli”
”Issueing of <PATHNAME> failed du to a communication error”
ERR DGEN ”Communication error on Issue TC/stimuli”
”Issueing of <PATHNAME> failed du to a timeout situation. The timeout value (currently <NNNN> ms) is an input value to the issue command”
ERR DGEN ”Could not make GDU.”
”Enditem has not been authorized. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU.”
”Enditem is disabled. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed, SAS not connected”
”SAS not connected for GDU. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed, NACK received”
”SAS has send a NACK for GDU. Pathname: <PATHNAME>”
ERR DGEN ”Issuing of GDU failed”
”Issuing of GDU failed due to unexpected failure. Pathname: <PATHNAME>”
ERR DGEN ”Error in sending a message to an AP.”
”Application <SAS_NAME> tried to send a message to AP <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for raw value.”
”Application <SAS_NAME> tried to read the raw value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for engineering value.”
”Application <SAS_NAME> tried to read the eng. value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Error in request for engineering value.”
”Application <SAS_NAME> tried to write the eng. value of <PATHNAME>, but this pathname could not be translated to an SID. ”
ERR DGEN ”Initialisation Error”
”Error during initialisation of TES local datapool”
ERR DGEN ”Could not enable issueing”
”Simulation is suspended. Could not enable issueing.”
ERR DGEN ”Cannot enable GDU”
”Could not enable enditem <PATHNAME>”
ERR DGEN ”Could not disable issueing”
”Simulation is suspended. Could not disable issueing.”
ERR DGEN ”Cannot disable GDU”
”Could not disable enditem <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Incorrect parameter type definition. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Wrong number of parameters to a gdu list. Pathname: <PATHNAME>”

Copyright per DIN 34

ERR DGEN ”Could not make GDU list”
”Incorrect parameter type definition. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU list”
”Index parameters for GDU list out of allowed ranges. Pathname: <PATHNAME>”
ERR DGEN ”Could not announce GDU service”
”Simulation is suspended. Could not announce the GDU service.”
ERR DGEN ”Could not withdraw GDU service”
”Simulation is suspended. Could not withdraw the GDU service.”
ERR DGEN ”Could not issue GDU”
”Simulation is suspended. Could not issue GDU : <PATHNAME>”
ERR DGEN ”Could not make GDU”
”GDU service not announced. Pathname: <PATHNAME>”
ERR DGEN ”Error during issue of GDU list”
”GDU list <PATHNAME>. Errors detailled in earlier messages”
ERR DGEN ”Could not make GDU.”
”The enditem is unknown. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU.”
”The enditem is not a local GDU. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong number of parameters. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong parameter type <TYPE> specified for GDU at position <NNNN>
Parameter type <UNSIGNED INTEGER/INTEGER/REAL/TIME/STATE CODE/
PATHNAME> expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong parameter type specified for GDU at position <NNNN>
Parameter type <UNSIGNED INTEGER/INTEGER/REAL/STRING/TIME/
STATE CODE/PATHNAME> expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong raw value type specified in MDB. Engineering values of types integer
can only be decalibrated to integer and unsigned integer raw values.
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Wrong raw value type specified in MDB. Engineering values of types real
can only be decalibrated to float, integer and unsigned integer raw values.
Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert the real
parameter. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert the actual
string value. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”

Copyright per DIN 34

”Time parameter not defined (date part not set). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Time parameter has to be greater than the TAI epoch specified in the TES configuration file Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert a time value (40 bits needed). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Size of GDU parameter at position <NNNN> does not allow to insert a statecode value (64 bits needed). Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Unsupported parameter type <TYPE> specified for parameter number : <NNNN> Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Unsupported parameter type specified for parameter number : <NNNN> Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Length error for parameter number <NNNN>. Value exceeds length of parameter or
parameter exceeds the CCSDS packet length. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Length error for parameter number <NNNN>. Value exceeds length of parameter or
parameter exceeds the binary packet length. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Onboard execution time contains a time but no date. Date required. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Onboard execution time has to be greater than the TAI epoch specified in the TES configuration file Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”GDU service not announced Pathname: <PATHNAME>”
— The addressed SAS did not anounce GDU service yet
ERR DGEN ”Could not make GDU”
”Incorrect default value. Real type expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Analog stimuli requires float parameters Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Incorrect default value. State code type expected. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”The parameter is not a discrete value Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Enditem is disabled Pathname: <PATHNAME>”

Copyright per DIN 34

ERR DGEN ”Could not make GDU”
”The decalibration parameter was out of range. Please check allowed range and types for this TC/stimuli in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”The decalibration definition is invalid. Please check the decalibration definition in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Invalid calibration type defined in MDB. Please check raw and engineering types for this TC/stimuli in MDB. Pathname: <PATHNAME>”
—> A decalibration was defined in the DB for the GDU.
—> The type defined there does not
—> match the type of the parameter given for the GDU
ERR DGEN ”Could not make GDU”
”Missing default value to TC/stimuli. Please provide a value as parameter or assign a default value in MDB. Pathname: <PATHNAME>”
ERR DGEN ”Could not make GDU”
”Exception <EXCEPTION_NAME> was raised when construction a GDU. Pathname: <PATHNAME>”
ERR DGEN ”Cannot set SAS name for GDU”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”
ERR DGEN ”Negative TAI epoch time”
”Current time has to be greater than the TAI epoch specified in the TES configuration file”
ERR DGEN ”Negative execution time”
”Onboard execution time has to be greater than the TAI epoch specified in the TES configuration file”
ERR DGEN ”Current mode wrong.”
”Set CCSDS end point not allowed in the current mode.”
ERR DGEN ”Invalid node name”
”The given pathname is not a CCSDS end point.”
ERR DGEN ”Current mode wrong.”
”Enable of SW command not allowed in the current mode.”
ERR DGEN ”Current mode wrong.”
”Disable of SW command not allowed in the current mode.”
ERR DGEN ”SW command for FLAP not found.”
”SW command <PATHNAME> for FLAP <PATHNAME> not found”
ERR DGEN ”SW command not found.”
”SW command <PATHNAME> not found”
ERR DGEN ”SW command disabled.”
”SW command <PATHNAME> disabled.”
ERR DGEN ”GDU service not announced.”
”The application <SAS_NAME> has not annonunced its GDU service.”

Copyright per DIN 34

ERR DGEN ”ADU service not announced.”
”The application <SAS_NAME> has not annonunced its ADU service.”
ERR DGEN ”Ground Node not specified.”
”The Ground Node parameters was not given and the default Ground Node is not set.”
ERR DGEN ”Invalid node parameter.”
”Application id for the node pair (ground node: <PATHNAME> to onboard node: <PATHNAME>) could not be found.”
ERR DGEN ”Invalid node parameters.”
”Application id’s for the node pair (ground node: <PATHNAME> to onboard node: <PATHNAME>) could not be found.”
ERR DGEN ”Sending of SW command failed.”
”Sending of the SW command <PATHNAME> failed.”
ERR DGEN ”Set Device address failed”
”No APID is defined for end point pair : source <PATHNAME> / destination <PATHNAME>”
ERR DGEN ”Cannot set SAS name for SWOP”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”
ERR DGEN ”Cannot set SAS name for Response Packet”
”Parameter OLD_SAS_NAME does not match the current SAS name for <PATHNAME>”

WRN DGEN ”Warning : incomplete CCSDS packet”
”Value missing for time parameter. Parameter will not be set in the GDU.
Pathname : <PATHNAME>”
WRN DGEN ”Warning : incomplete GDU – CCSDS packet”
”CCSDS packet without secondary header. Parameter ONBOARD_EXECUTION_TIME
will not be set in the GDU. Pathname : <PATHNAME>”
WRN DGEN ”Parameter list truncated for SW command”
”The parameter part of the software command <PATHNAME> only contains <NNNN> Bytes.
Not all IN parameters have been inserted.”
WRN DGEN ”Response packet too short”
”The OUT parameter part of the response packet for <PATHNAME> only contains <NNNN>
Bytes. Remaining OUT parameters are set to default value.”

INFO DGEN ”ENABLE_ENDITEM ignored”
”UCL system library routine ENABLE_ENDITEM ignored in replay mode.”
INFO DGEN ”Enabled enditem”
”Enabled issuing enditem with pathname: <PATHNAME>”
INFO DGEN ”Enabled enditems”

Copyright per DIN 34

”Enabled issuing of <NNNN> enditems contained in : <PATHNAME>”
INFO DGEN ”DISABLE_ENDITEM ignored”
”UCL system library routine DISABLE_ENDITEM ignored in replay mode.”
INFO DGEN ”Disabled enditem”
”Disabled issuing enditem with pathname: <PATHNAME>”
INFO DGEN ”Disabled enditems”
”Disabled issuing of <NNNN> enditems contained in : <PATHNAME>”
INFO DGEN ”GDU Service Announced”
”The application <SAS_NAME> has announced its GDU Service.”
INFO DGEN ”GDU Service Withdrawn”
”The application <SAS_NAME> has withdrawn its GDU Service.”
INFO DGEN ”ISSUE ignored”
”UCL system library routine ISSUE ignored in replay mode.”
INFO DGEN ”New SAS reference for GDU”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s referenced in <PATH-NAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s referenced in <PATH-NAME>
where the SAS reference was <SAS_NAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s under <PATHNAME>”
INFO DGEN ”New SAS reference for GDU’s”
”SAS reference has been set to <SAS_NAME> for all GDU’s under <PATHNAME>
where the SAS reference was <SAS_NAME>”
INFO DGEN ”Send GDU in sim mode”
”<PATHNAME>”
INFO DGEN ”Single GDU issued”
”GDU <PATHNAME> issued”
INFO DGEN ”List of GDU packets issued”
”Packet list <PATHNAME> issued”
INFO DGEN ”SW command sent.”
”SW command <PATHNAME> sent, transaction id=<HHHH> where <HHHH> is the
hexadecimal of the CCSDS primary header of the telecommand”
INFO DGEN ”Device address changed for APID <NNNN>”
”New device address is <STRING>”
INFO DGEN ”New SAS reference for SWOP”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”
INFO DGEN ”New SAS reference for Response Packet”
”SAS reference has been set to <SAS_NAME> for <PATHNAME>”

Copyright per DIN 34

INFO DGEN ”New SAS reference for SWOP items”
”SAS reference has been set to <SAS_NAME> for all SWOP’s and response packets under <PATHNAME>”
INFO DGEN ”New SAS reference for SWOP items”
”SAS reference has been set to <SAS_NAME> for all SWOP’s and response packets under <PATHNAME> where the SAS reference was <SAS_NAME>”

TYPE GROUP Text

ERR DGVF ”TC Verification failed”
”The verification of <PATHNAME> failed : <NNNN> check(s) failed.”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : enditem <PATHNAME> is not acquired”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : check of enditem <PATHNAME> failed – value is invalid”
ERR DGVF ”TC Verification check failed”
”Command <PATHNAME> : check of enditem <PATHNAME> failed – last value: <VALUE>”
ERR DGVF ”Could not get GDU verification status”
”Simulation is suspended. Could not GDU verification status : <PATHNAME>”
ERR DGVF ”Could not get GDU verification status.”
”The enditem is unknown. Pathname: <PATHNAME>”
ERR DGVF ”Could not get GDU verification status.”
”The enditem is not a local GDU. Pathname: <PATHNAME>”

WRN DGVF ”TC Verification aborted”
”TC verification for <PATHNAME> is aborted due to the stop of the test node.”
WRN DGVF ”TC Verification aborted”
”Command <PATHNAME> has been issued while the verification part of its last generation
has not been completed. Previous TC Verification is aborted”

INFO DGVF ”TC Verification successful”
”The verification of <PATHNAME> is successful.”
INFO DGVF ”GET_VERIFICATION_STATUS ignored”
”UCL system library routine GET_VERIFICATION_STATUS ignored in replay mode.”

TYPE GROUP Text

ERR HCI ”Shutdown Error.”
”The status <STATUS> was received when disconnecting from HCI.”

TYPE GROUP Text

Copyright per DIN 34

MSG LOG ”<AP_TEXT>”
”<AP_TEXT>”

TYPE GROUP Text

ERR MON ”Calibration Error”
”Calibration of <TYPE> to <TYPE> not supported for measurement <PATH-NAME>.”
ERR MON ”Calibration Error”
”Raw or calibrated value out of defined ranges for measurement <PATHNAME>.”
ERR MON ”Calibration Error”
”Calibration description not properly defined for measurement <PATHNAME>.”
=> exceptions occured in CALIBRATION package
ERR MON ”Calibration Error”
”Enditem corrupted: Could not calibrate for measurement <PATHNAME>.”
=> in CALIBRATE_RAW_VALUE if unknown exception occurs
ERR MON ”Limit not set in Enable_monitoring”
”Limit set no. <NNNN> is undefined for enditem <PATHNAME>.”
ERR MON ”Change limits failed.”
”<PATHNAME> is not a measurement, sw variable or derived value”
ERR MON ”Change limits failed.”
”<PATHNAME> is not known.”
ERR MON ”Change limits failed.”
”<PATHNAME> has no monitoring limits.”
ERR MON ”Set nominal limit failed.”
”limit value outside integer range for <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”Can not set nominal limits for nonscalar value <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”New limit value inconsistent with the current limit set of <PATHNAME>”
ERR MON ”Set nominal limit failed.”
”Can not set nominal limits for non integer value <PATHNAME>”
ERR MON ”Set expected state code failed”
”<PATHNAME> is not a state code type enditem.”
ERR MON ”Failed to set expected string.”
”<PATHNAME> is not a string type enditem.”
ERR MON ”Set limit set failed.”
”The limit set <NNNN> is undefined for enditem <PATHNAME>”
ERR MON ”Set exception count failed”
”Exception count for <PATHNAME> not set (provided value too large.)”
ERR MON ”Invalid limits set.”
”The limit set number <NNNN> is illegal”

Copyright per DIN 34

ERR MON ”Invalid delta limit.”
”The delta limit can not be negative.”
ERR MON ”Invalid exception count limit.”
”The exception count must be greater than zero.”
ERR MON ”Emergency AP not found.”
”Emergency AP <PATHNAME> was not loaded initially. The AP will be loaded from MDB.”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it cannot be loaded from the MDB”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it has parameters”
ERR MON ”Could not start emergency AP.”
”AP <PATHNAME> should have been started as a result of a monitoring exception, but it is already running”

INFO MON ”Enabling monitoring.”
”Enabling monitoring with limit set <NNNN> for enditem(s) <PATHNAME>”
INFO MON ”Disabling monitoring.”
”Disabling monitoring for enditem(s) <PATHNAME>”
INFO MON ”New nominal limit definition.”
”Nominal high limit for <PATHNAME> has been set.”
INFO MON ”New nominal limit definition.”
”Nominal low limit for <PATHNAME> has been set.”
INFO MON ”New nominal limit definition.”
”Nominal delta limit for <PATHNAME> has been set.”
INFO MON ”New expected state code.”
”Expected state code for <PATHNAME> has changed to <STATCODE>.”
INFO MON ”New expected Value”
”Expected value for <PATHNAME> has changed to <STRING>.”
INFO MON ”Setting new limit set.”
”Setting limit set <NNNN> for enditem(s) <PATHNAME>”
INFO MON ”New Exception Count”
”Exception Count for <PATHNAME> has changed to <NNNN>.”

TYPE GROUP Text

ERR REPL ”Replay initialisation error.”
”Replay initialisation error. Replayer must be initialised in replay–mode. Replaying stopped.”
ERR REPL ”Replay initialisation error.”
”Replay initialisation error. The replay end time is before the start time.

Copyright per DIN 34

Replaying stopped."
ERR REPL "Replay initialisation error."
"Replay initialisation error. The replay start and end time is equal.
Replaying stopped."
ERR REPL "Replay initialisation error."
"Replay initialisation error. Can not read archive file: <FILENAME>
Replaying stopped."
ERR REPL "End of archive file(s) reached."
"End of archive file(s) reached. Replaying stopped."
ERR REPL "Exception raised."
"Exception DURATION_OVERFLOW raised due to too long wait period
between two archive file items. Replaying stopped."
ERR REPL "Replaying stopped."
"Replaying stopped, because an unexpected exception was raised."
ERR REPL "Replay initialisation error."
"Exception DURATION_OVERFLOW raised due to too long wait period for
the first item to be replayed."
ERR REPL "Wrong TES mode."
"Wrong TES mode. Not allowed to restart TES in replay mode without reinitialising."

WRN REPL "TES is late in replay."
"TES is more than 5 seconds late in replay Possible cause: too many
data or speed too high"
WRN REPL "Nothing to replay"
"If MTP, cannot set local time and SMT value. If test with several test nodes,
synchronisation will not be achieved. Use local time setup instead"
WRN REPL "replay with non running SMT"
"If test with several test nodes, synchronisation will not be achieved.
Use local time setup instead"
WRN REPL "Nothing to replay"
"If MTP, cannot set local time and SMT value. If test with several test nodes,
synchronisation will not be achieved"
WRN REPL "Nothing to replay"
""

INFO REPL "Replaying stopped by the user."
"Replaying stopped by the user. Replaying stopped."
INFO REPL "Replaying of data stopped."
"Replaying of data stopped, because the end of the last archive
file was reached. Replaying stopped."
INFO REPL "Replaying of last item done"
"Waiting for replay end time"
INFO REPL "Nothing to replay"
""

Copyright per DIN 34

INFO REPL ”Initial delay : <HH> hrs <MI> min <SS> sec.”
”The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>. Consider a reduction of the time frame to replay if the initial delay appears to be too long.”
INFO REPL ”Initial delay : <MI> min <SS> sec.”
”The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>. Consider a reduction of the time frame to replay if the initial delay appears to be too long.”
INFO REPL ”Initial delay : <SS> sec.”
”The first data to replay has been found in the archive at local time <DD>.<MM>.<YYYY> <HH>:<MI>:<SS>.”
INFO REPL ”Waiting for replay end time”
”Waiting for replay end time”
INFO REPL ”Replayed <GDU_TYPE> command”
”Replayed command <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT].”
INFO REPL ”Started acquisition of a data packet”
”Packet <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT].”
INFO REPL ”Stopped acquisition of a data packet”
”Packet <PATHNAME>, orig. time was <TIME> [LT] & <TIME> [SMT].”
INFO REPL ”Suspended replaying.”
”Suspended replaying on <NODE>”
INFO REPL ”Resumed replaying.”
”Resumed replaying on <NODE>”

TYPE GROUP Text

ERR SAS ”acknowledge code out of range”,
”The acknowledge code delivered by SAS <NAME> is out of range.”
ERR SAS ”Link state control error”
”SAS name unknown”
ERR SAS ”Not connected”
”Application name: <SAS_NAME>”
ERR SAS ”Sending a command to application failed”
”The command <COMMAND> did not get acknowledged or was not read by <SAS_NAME> within the timeout period of: <NNNN> ms”
ERR SAS ”Error in communication with application”
”Sending <COMMAND> to application <SAS_NAME> returned with status: <STATUS>”
ERR SAS ”Multiple load”
”Application <SAS_NAME> already running”

Copyright per DIN 34

ERR SAS ”OS_BINDING error”
”Error: Create_Process failed, file <FILENAME> not found”
=> SAS could not be started. Executable not found
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. Host name not known on this node”
=> SAS could not be started. Host not found
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. The server process was not found on the indicated host”
=> The process_creation_server is not running or not responding
(time–out)
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>. The server process was killed”
=> The process_creation_server is not running or crashed
ERR SAS ”OS_BINDING error”
”Error: Create_Process failed for file <FILENAME>”
=> an exception occured when calling OS_BINDING
ERR SAS ”OS_BINDING error”
”Exception received in call to OS_BINDING”
ERR SAS ”Can not connect”
”Application name : <SAS_NAME>”
=> Application connected, but could not be registered internally
(P_USER_REGISTER.CONNECT_SAS)
ERR SAS ”SAS is not loaded”
”Loaded application name : <SAS_NAME> Connecting application name : <SAS_NAME>”
ERR SAS ”Time out of Load_application”
”Application name : <SAS_NAME> did not connect within the the timeout period <NNNN> seconds (defined by the config variable TES.LOAD_APPLICATION_TIMEOUT).”
ERR SAS ”Time out of Load_application”
”Application name : <SAS_NAME> did not connect within the the timeout period <NNNN> seconds (defined by the config variable TES.LOAD_APPLICATION_TIMEOUT).
Please note that manual loading of SASes is enabled.”
ERR SAS ”Connect_SAS called out of sequence”
”Application name : <SAS_NAME>”
ERR SAS ”Mode error”
”Connect_SAS called in illegal mode”
ERR SAS ”Mode error”

Copyright per DIN 34

”Disconnect_SAS illegal in current mode”

ERR SAS ”Application not connected”
”Application ID: <NNNN>”

ERR SAS ”Mode error”
”Send_error_message illegal in current mode”

ERR SAS ”Mode error”
”Get_SAS_status_ID illegal in current mode”

ERR SAS ”Get SAS status ID”
”Not connected: <NAME>”

ERR SAS ”Get SAS status ID”
”Invalid name: <NAME>”

INFO SAS ”SAS disconnected due to stop of TES”
”<SAS_NAME>”

<XXX> SAS ”<SAS_TEXT1>”
”<SAS_TEXT2>”

TYPE GROUP Text

ERR TES ”STORAGE_ERROR raised!”
”A STORAGE_ERROR exception was raised in SW Unit <UNIT_NAME> .
Please adjust the storage size for the task affected”

ERR TES ”Exception raised!”
”Exception: <EXCEPTION_NAME> was raised in SW Unit: <UNIT_NAME>”

ERR TES ”Coded Message Error”
”The provided sid does not correspond to a user message definition.”

ERR TES ”AP Communicator Error”
”TES internal error : <EXCEPTION_NAME> occurred.”

ERR TES ”Initial setup failure.”
”TES internal error : <EXCEPTION_NAME> occurred.”

ERR TES ”Initial setup failure”
”TES internal error : <EXCEPTION_NAME> occurred.”

ERR TES ”AP Controller Error”
”TES internal error : <EXCEPTION_NAME> occurred.”

ERR TES ”Sending message failed”
”Return status from General Comms: <STATUS>”

ERR TES ”Initialisation Error”
”Invalid mode in the INIT operation.”

ERR TES ”Start Error”
”Need initialisation before START.”

ERR TES ”Start Error”
”Invalid mode in the START operation.”

ERR TES ”Stop Error”
”Invalid mode in the STOP operation.”

Copyright per DIN 34

ERR TES ”TSS Error”
”<TEXT>”
Already member of another SMT–domain.
—> Has the TSS process on the test node already been initialised with
—> another configuration, defining a different SMT domain (i.e. e.g. having
—> a different MTP)
—> Stop whole test configuration and start again
_____ Issuing node not SMT–server for given domain.
Return code = TS_OPS_TYPES.ETSNOTSMTSRV
—> was the TSS server (TSP) not setup correctly before ?
—> Setup the whole test configuration again using TSCV
—> If not successful: restart TSP process and try again
ERR TES ”Simulator received illegal command.”
””
ERR TES ”Calibrated value log failure”
”Logging of <ENG_VALUE_TYPE> items not supported, for <PATHNAME>.”
ERR TES ”Data type conversion failure”
”Type <ENG_TYPE> could not be converted, for <PATHNAME>”
ERR TES ”Data type conversion failure”
”Type <RAW_TYPE> could not be converted, for <PATHNAME>”
ERR TES ”Could not insert SID in local SID table”
”Not enough memory could be found to create the local SID table for data distribution”
ERR TES ”Error Message Loading Failure.”
”The status <STATUS> was received when loading coded error message definitions.”
ERR TES ”Announce enditems to HCI failed”
”Logging to HCI might fail but proceeding with setup”
ERR TES ”Init kernel failed”
”The return status was <STATUS>”
ERR TES ”Init Clock failed”
”Check if tss is running. The return status was <STATUS>”
ERR TES ”Init Archive failed”
”The return status was <STATUS>”
ERR TES ”Init UCLI failed”
”The return status was <STATUS>”
ERR TES ”Init CM failed”
”The return status was <STATUS>”
ERR TES ”Init GDU manager failed”
”The return status was <STATUS>”
ERR TES ”Init Simulator failed”
”The return status was <STATUS>”
ERR TES ”Init Replayer failed”

Copyright per DIN 34

”The return status was <STATUS>”
ERR TES ”Error communicating to remote TES”
”cannot inform remote TES <TES_NAME> about completion of TC Verification”
ERR TES ”Error during setup of communication”
”Return status from ADT_SYSTEM_TOPOLOGY.WHO_AM_I: <STATUS>”
ERR TES ”Error reading message : <STATUS>”
”Error reading message : <STATUS>”
ERR TES ”Initialisation error.”
”Initialisation error. Simulator must be initialised in simulation–mode.”
ERR TES ”Could not stop Simulator properly.”
”TES internal error : Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”TES internal error.”
”TES internal error : Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”Simulation–data corrupted.”
”Simulation–data corrupted. Measurement <PATHNAME>) in ADU (<PATHNAME>)
already exists.”
ERR TES ”Could not start simulate ADU.”
”Could not start simulate ADU <PATHNAME>. Could not get a new ADU_Generator from the ADU_Generator_Pool.”
ERR TES ”Could not stop simulation of ADU.”
”TES internal error : Could not stop ADU <PATHNAME> properly. Could not put ADU_Generator into ADU_Generator_Pool.”
ERR TES ”Initial setup failure.”
”TES internal error : Could not create datastructure.”
=> Error when creating a message buffer for AP/SAS messages
=> or Error when creating a buffer for APs (U_WORD_STORAGE)
=> ADT_SID_TO_GENERATOR_MAP.CREATE_ERROR
ERR TES ”Initialisation Error.”
”Could not initialise the PI Server of the TES Core.”
ERR TES ”Start Error.”
”Could not start the Clock manager.”
ERR TES ”Start Error.”
”Could not start the Archive manager.”
ERR TES ”Start Error.”
”Could not start the User manager.”
ERR TES ”Start Error.”
”Could not start the Kernel.”
ERR TES ”Start Error.”
”Could not start the GDU manager.”
ERR TES ”Start Error.”

Copyright per DIN 34

```
          ”Could not start the Data Dispatch Service.”
ERR   TES     ”Start Error.”
          ”Could not start the UCL manager.”
ERR   TES     ”Start Error.”
          ”Could not start the SW Command manager.”
ERR   TES     ”Start Error.”
          ”Could not start the Simulation manager.”
ERR   TES     ”Start Error.”
          ”Could not start the Replayer.”
ERR   TES     ”Start Error.”
          ”Could not start the TES Core.”
ERR   TES     ”Start Error.”
          ”Could not start the PI Server of the TES Core.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Simulation manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Replayer.”
ERR   TES     ”Stop Error.”
          ”Could not stop the UCL manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Data Dispatch Service.”
ERR   TES     ”Stop Error.”
          ”Could not stop the GDU manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the SW cmd manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Kernel.”
ERR   TES     ”Stop Error.”
          ”Could not stop the User manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Archive manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the Clock manager.”
ERR   TES     ”Stop Error.”
          ”Could not stop the PI Server of the TES Core.”
ERR   TES     ”Shutdown Error.”
          ”Could not shut down the PI Server of the TES Core.”
ERR   TES     ”Initialisation Error.”
          ”Initialisation only allowed in Available and Idle modes and
           when shutdown is not in progress”
ERR   TES     ”Initialisation Error.”
        ”Initialisation when in Available mode must be with forced loading of MDB data.”
ERR   TES     ”Initialisation Error.”
```

Copyright per DIN 34

”Wrong Active Mode provided in Init (Use Normal, Simulation or Replay).”
ERR TES ”Initialisation Error.”
”Can not find TES instance name in System Topology Table.”
ERR TES ”Initialisation Error.”
”Could not stop TES Core Idle mode.”
ERR TES ”Initialisation Error.”
”Could not initialise the TES Core.”
ERR TES ”Initialisation Error.”
”Could not start TES Core to Idle mode.”
ERR TES ”Start Error.”
”Start not allowed in current mode.”
ERR TES ”Stop Error.”
”Stop not allowed in current mode.”
ERR TES ”Stop Error.”
”Could not stop the TES Core (Fatal Error).”
ERR TES ”Stop Error.”
”Could not stop the TES Core.”
ERR TES ”Suspend Error.”
”Current mode is <MODE> . Suspend is only meaningful in SIMULATION and REPLAY mode.”
ERR TES ”Suspend Error.”
””
ERR TES ”Resume Error.”
”Current mode is <MODE>. Resume is only meaningful in SIMULATION and REPLAY mode.”
ERR TES ”Resume Error.”
”Could not resume simulation.”
ERR TES ”Resume Error.”
”Could not resume replaying.”
ERR TES ”Shutdown Error.”
”Shutdown not allowed in current mode.”
ERR TES ”Shutdown Error.”
”Could not stop the TES Core.”
ERR TES ”Synchronize Error.”
”Synchronize not allowed in current mode.”
ERR TES ”Synchronize Error.”
”Synchronization not possible for equal client and server.”
ERR TES ”Synchronize Error.”
”Could not synchronize with remote TES : <TES_NAME>”
ERR TES ”Connect error”
”Could not connect with remote TES : <TES_NAME>”
ERR TES ”Operation not Available”
”TES must be shutdown first!”

Copyright per DIN 34

ERR TES ”Could not read HK–values from TSS.”
”<TEXT>”
ERR TES ”Wrong user calling”
”Only HCI and TSCV can call this operation, not : <NAME>”
ERR TES ”HCI/TSCV not connected”
”HCI/TSCV name : <NAME>”
ERR TES ”Name and ID conflict”
”Name: <NAME> and ID: <NNNN> are not connected in CM”
ERR TES ”Operation not allowed in <MODE> mode”
”Operation <OPERATION>”
ERR TES ”TES internal error.”
”TES internal error : No free AP info map entry.”
ERR TES ”Could not store message.”
”Could not store message from AP–ID <NNNN> to AP–ID <NNNN>. Not enough memory.”
ERR TES ”Could not store message.”
”Could not store message from Application with ID <NNNN> to AP–ID <NNNN>. Not enough memory.”
ERR TES ”Get Message Error”
”TES internal error : <EXCEPTION_NAME> occurred.”

WRN TES ”CLOSE_ARCHIVE ignored”
”UCL system library routine CLOSE_ARCHIVE ignored in replay mode.”
WRN TES ”ENABLE_ARCHIVING ignored”
”UCL system library routine ENABLE_ARCHIVING ignored in replay mode.”
WRN TES ”DISABLE_ARCHIVING ignored”
”UCL system library routine DISABLE_ARCHIVING ignored in replay mode.”
WRN TES ”START_SMT ignored”
”UCL system library routine START_SMT ignored in replay mode.”
WRN TES ”STOP_SMT ignored”
”UCL system library routine STOP_SMT ignored in replay mode.”
WRN TES ”Trying to connect null name”
””
WRN TES ”Application cannot connect”
”Application <NAME> not connected, max users connected”

INFO TES ”Connected to Current CCU.”
””
INFO TES ”Connected to HCI.”
””
INFO TES ”Connection to HCI failed.”
”Status= <STATUS>”
INFO TES ”Disconnecting from HCI.”
””

Copyright per DIN 34

INFO TES ”Connected to TRDB (DBS).”
””
INFO TES ”Disconnected from TRDB (DBS).”
””
INFO TES ”Stopped DBS RPI.”
””
INFO TES ”Disconnected from HCI.”
””
INFO TES ”Entering ERROR mode.”
””
INFO TES ”Entering AVAILABLE mode.”
””
INFO TES ”Initialisation parameters.”
”Mode: <MODE>
MTP: <TRUE/FALSE>
Load from DB: <TRUE/FALSE>
CCU: <PATHNAME>
Measurements: <NNNN>
ADUs: <NNNN>
GDUs: <NNNN>
GDU lists: <NNNN>
Mon. lists: <NNNN>
APs: <NNNN>
User libs: <NNNN>
EGSE nodes:<NNNN>
User MSGs: <NNNN>”
INFO TES ”Entering <MODE> mode.”
””
INFO TES ”Suspended simulation.”
”Suspended simulation on <NODE>”
INFO TES ”Resumed simulation.”
”Resumed simulation on <NODE>”
INFO TES ”Shutting down now ...”
””
INFO TES ”Application connected”
”Application name : <NAME>”
INFO TES ”User disconnected <SAS_NAME>”
””
INFO TES ”HCI/TSCV connected”
”Name : <NAME>. This user was already connected.”
INFO TES ”HCI/TSCV connected”
”Name : <NAME>”
INFO TES ”HCI/TSCV disconnected”

Copyright per DIN 34

”Name : <NAME>”

MSG TES ”<TEXT1>”
”Trace: <TEXT2>”

TYPE GROUP Text

ERR TRDB ”Failed to log engineering value.”
”Logging of the engineering value <PATHNAME> failed with the status <STATUS>”
ERR TRDB ”Initialisation Error.”
”The status <STATUS> was received when connecting to TRDB (DBS).”
ERR TRDB ”Shutdown Error.”
”The status <STATUS> was received when disconnecting from TRDB (DBS).”
ERR TRDB ”Shutdown Error.”
”The status <STATUS> was received when stopping the DBS RPI.”

INFO TRDB ”EVL status changed.”
”The engineering value logging for <PATHNAME> is turned <ON/OFF>”

TYPE GROUP Text

ERR UCLI ”Module in Use”
”Attempted to overwrite an I–Code module which is in use. Automated Procedure :
<PATHNAME>”
ERR UCLI ”Module in Use”
”Attempted to overwrite an I–Code module which is in use. User Library : <PATHNAME>”
ERR UCLI ”Automated Procedure not found”
”File for Automated Procedure <PATHNAME> could not be loaded.”
ERR UCLI ”I–code for Automated Procedure not found”
”Automated Procedure <PATHNAME> could not be loaded due to missing I–code (not compiled?)”
ERR UCLI ”User Library not found”
”File for User Library <PATHNAME> could not be loaded.”
ERR UCLI ”I–code for User Library not found”
”User Library <PATHNAME> could not be loaded due to missing I–code (not compiled?)”
ERR UCLI ”Internal error.”
”Time_Slicer: Removed too many clients.”
ERR UCLI ”AP initialisation failure.”
”Could not initialise AP <PATHNAME>”
ERR UCLI ”Initialisation error”
”Could not start AP: <PATHNAME>. User library <PATHNAME> could not be

Copyright per DIN 34

loaded”
ERR UCLI ”Initialisation error”
”Could not execute user library routine (invalid code).”
ERR UCLI ”Initialisation error”
”Could not start AP–ID <NN>; could not access AP I–Code.”
ERR UCLI ”Initialisation error”
”Could not start AP <PATHNAME>. An error occurred during loading of I–code into the stack machine data structures.””
ERR UCLI ”Initialisation error”
”Could not start AP : <PATHNAME>. <EXCEPTION_NAME> occurred.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. Error during the execution of a system library routine.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library

<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error during the execution of a system library routine.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. An unexpected exception occured during execution.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library

<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. An unexpected exception occured during execution.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library

<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. This instruction is illegal in MULTI mode.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library

Copyright per DIN 34

<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>.
This instruction is illegal in MULTI mode.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. Constraint– or numeric error raised.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library
<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>.
Constraint– or numeric error raised.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. ~:~ (’no time’) used as a time operand.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library
<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>.
~:~ (’no time’) used as a time operand.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. Illegal operator.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library
<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>.
Illegal operator.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. Error when accessing the TES local database.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library

<PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>.
Error when accessing the TES local database.”

ERR UCLI ”TES Internal error : Execution error”
”Stopped execution of AP: <PATHNAME>. The current stack machine status is:
<OK/ERROR/HALT>”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. This is an illegal instruction at this point.”

ERR UCLI ”Execution error”

Copyright per DIN 34

”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. This is an illegal instruction at this point.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. A range or index check resulted in an error.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. A range or index check resulted in an error.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Stack overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory underflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory overflow.”

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Memory overflow.”

Copyright per DIN 34

ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. The PC is outside the code frame.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. The PC is outside the code frame.”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. Error trap; the error code was: <NNNN>”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. Error trap; the error code was: <NNNN>”
ERR UCLI ”Execution error”
”Stopped execution of AP: <PATHNAME>. FATAL : An exception occurred in the
ERROR_TRAP routine.”
ERR UCLI ”Execution error”
”AP: <PATHNAME>. Could not store string read from the TES local database; not enough
space was allocated for the string. SID: <NNNN>”
ERR UCLI ”Execution error”
”AP: <PATHNAME>. Could not write string to the TES local database (too long). SID: <NNNN>”
ERR UCLI ”Execution error”
”AP: <PATHNAME>, Instruction : <INSTRUCT>. The duration resulting from the subtraction
could not be handled by the current Ada implementation.”
ERR UCLI ”Termination error”
”Could not terminate AP properly: <PATHNAME>. An error occurred when returning user
library parameters to HCI.”
ERR UCLI ”Termination error”
”Could not terminate AP properly: <PATHNAME>. <EXCEPTION_NAME> occurred.”
ERR UCLI ”Could not open file for debug output”
”For Debug–Option ’<XXXXX>’. An exception occurred. Execution continues with
debugging switched off.”
ERR UCLI ”Derived value error”

Copyright per DIN 34

”<PATHNAME> : Error during the execution of a system library routine.”
ERR UCLI ”Derived value error”
”<PATHNAME> : An unexpected exception occured during execution.”
ERR UCLI ”Derived value error”
”<PATHNAME> : This instruction is illegal in DOUBLE mode.”
ERR UCLI ”Derived value error”
”<PATHNAME> : This instruction is illegal in MULTI mode.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Constraint– or numeric error raised.”
ERR UCLI ”Derived value error”
”<PATHNAME> :~:~ (’no time’) used as a time operand.”
ERR UCLI ”TES Internal error : Derived value error”
”<PATHNAME> : Illegal operator.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Error when accessing the TES local database.”
ERR UCLI ”TES Internal error : Derived value error”
”Stopped execution of AP: <PATHNAME>. The current stack machine status is: <OK/ERROR/HALT>”
ERR UCLI ”Derived value error”
”<PATHNAME> : illegal instruction.”
ERR UCLI ”Derived value error”
”<PATHNAME> : A range or index check resulted in an error.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Stack underflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Stack overflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Memory underflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Memory overflow.”
ERR UCLI ”Derived value error”
”<PATHNAME> : The PC is outside the code frame.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Error trap; the error code was: <NNNN>”
ERR UCLI ”Derived value error”
”<PATHNAME> : FATAL : An exception occurred in the ERROR_TRAP routine.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not store string read from the TES local database; not enough space was allocated for the string. SID: <NNNN>”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not write string to the TES local database (too long). SID: <NNNN>”

Copyright per DIN 34

ERR UCLI ”Derived value error”
”<PATHNAME> : The duration resulting from the subtraction
could not be handled by the current Ada implementation.”
ERR UCLI ”Derived value error”
”<PATHNAME> : Could not start calculation”
ERR UCLI ”Error executing System Library routine”
”AP: <PATNAME>. System Library routine READ_MESSAGE_FROM_AP
raised <EXCEPTION_NAME>.”
ERR UCLI ”Error executing System Library routine”
”AP: <PATNAME>. System Library routine READ_MESSAGE_FROM_APPLICA-
TION
raised <EXCEPTION_NAME>.”
ERR UCLI ”Error in SW command parameter list”
”Too many parameters”
ERR UCLI ”Error in SW command parameter list”
”An illegal parameter type was found : <TYPE>”
ERR UCLI ”Error in SW command parameter list”
”An ””in out”” parameter was found in a parameter list.”
ERR UCLI ”Error in SW command parameter list”
”Parameter list too large: doesn’t fit into a CCSDS packet.”
ERR UCLI ”Error in GDU parameter list”
”Too many parameters”
ERR UCLI ”Error in GDU parameter list”
”An illegal one word scalar parameter type was found: <TYPE>”
ERR UCLI ”Error in GDU parameter list”
”An illegal two word scalar parameter type was found: <TYPE>”
ERR UCLI ”Error in GDU parameter list”
”An illegal parameter type was found: <TYPE>”
ERR UCLI ”Error in GDU parameter list”
”An illegal/incompatible scalar parameter type was found: <TYPE>”
ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to unknown System Library with library number
=<NNNN>”
ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to unknown System Library routine of the
<LIBRARY_NAME>; unknown procedure number =<NNNN>”
ERR UCLI ”Error executing System Library routine”
”Calling the System Library routine <ROUTINE_NAME> is not allowed from
HLCL.”
ERR UCLI ”Error executing System Library routine”
”Request from <NAME>. Call to System Library routine <ROUTINE_NAME>
returned with status <NNNN>”
ERR UCLI ”Error executing System Library routine”

Copyright per DIN 34

”Request from <NAME>. Call to System Library routine <ROUTINE_NAME> gave an unexpected exception”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to unknown System Library with library number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to unknown System Library routine of the <LIBRARY_NAME>; unknown procedure number =<NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to System Library routine <ROUTINE_NAME> returned with status <NNNN>”

ERR UCLI ”Error executing System Library routine”
”AP: <PATHNAME>. Call to System Library routine <ROUTINE_NAME> gave an unexpected exception”

ERR UCLI ”Error executing System Library routine”
”Calls to <LIBRARY_NAME> are not allowed for derived values.”

ERR UCLI ”Error executing System Library routine”
”SET_DEFAULT_WORKSTATION: DEFAULT_WORKSTATION is unchanged.

<HCI_NAME> is not a participated HCI in test configuration.”

ERR UCLI ”Error executing System Library routine”
”SET_CCSDS_APID: Invalid parameter range. The CCSDS application id is in range 0 .. 2047.”

ERR UCLI ”Internal Error”
”Couldn’t wait for AP because EXECUTE_AP status was <NNNN>”

ERR UCLI ”Internal Error”
”Error clearing data structures <PATHNAME>. <EXCEPTION_NAME> occurred.”

ERR UCLI ”HK Value Write Error”
”Error writing statement no.: <PATHNAME>. <EXCEPTION_NAME> occurred.”

ERR UCLI ”Error when calling HCI RPI”,
”Received the following return status when calling HCI_RPI.Return_UCL_user_lib_parameters : <STATUS>. The HCI instance addressed was <NAME>”

ERR UCLI ”UCLI is stopped”
”Could not start UCL user library routine. UCLI is stopped.”

ERR UCLI ”UCLI is stopped”
”Could not start <PATHNAME>. UCLI is stopped.”

ERR UCLI ”UCLI is stopped”
”Could not start AP. UCLI is stopped.”

ERR UCLI ”UCLI is stopped”
”Could not synchronise with AP, ID <NNNN>. UCLI is stopped.”

ERR UCLI ”UCLI is stopped”
”Can not terminate any APs. UCLI is stopped.”

Copyright per DIN 34

ERR UCLI ”UCLI is stopped”
”Could not terminate AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not suspend AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not resume AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not return AP identifier for <PATHNAME>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not write message to AP–ID <NNNN>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not write message to <PATHNAME>. UCLI is stopped.”
ERR UCLI ”UCLI is stopped”
”Could not return status of AP, ID <NNNN>. UCLI is stopped.”
ERR UCLI ”Could not start AP.”
”<PATHNAME> cannot be started because it has parameters but no parameter was provided”
ERR UCLI ”Could not start library routine.”
”Could not start UCL user library routine.The maximum number of APs is already running.”
ERR UCLI ”Could not start emergency AP.”
”Could not start <PATHNAME>. The maximum number of APs is already running.”
ERR UCLI ”Could not start AP.”
”Could not start <PATHNAME>. The maximum number of APs is already running.”
ERR UCLI ”AP not found.”
”Could not synchronise with AP. AP–ID <NNNN> not found.”
ERR UCLI ”AP not found.”
”Could not terminate AP, AP–ID <NNNN> not found.”
ERR UCLI ”Could not suspend AP.”
”Could not suspend <PATHNAME>. The AP has already been suspended on request.”
ERR UCLI ”AP not found.”
”Could not suspend AP, ID <NNNN>. AP not found.”
ERR UCLI ”Could not resume AP.”
”Could not resume <PATHNAME>. AP was not suspended on request.”
ERR UCLI ”AP not found.”
”Could not resume AP, ID <NNNN>. AP not found.”
ERR UCLI ”AP not found.”
”Could not write message to AP–ID <NNNN>; AP not found.”
ERR UCLI ”AP not found.”
”Could not write message to AP. <PATHNAME> not found.”
ERR UCLI ”Internal Error”
”Attempted to release the wrong Wait Table entry.”
ERR UCLI ”Failed to terminate AP(s).”

Copyright per DIN 34

”Failed to release wait events: <NNNN> out of <NNNN> external waits within the time–out period.”

ERR UCLI ”Failed to terminate AP(s).”
”Failed to terminate AP(s). <NNNN> APs out of <NNNN> confirmed termination within the time–out period.”

WRN UCLI ”Could not stop.”
”Could not stop: APs are running. FORCED STOP required.”

INFO UCLI ”Emergency AP started with AP_ID: <NNNN>”
”<PATHNAME> started with <PRIORITY> priority”
INFO UCLI ”AP started with AP_ID: <NNNN>”
”<PATHNAME> started with <PRIORITY> priority”
INFO UCLI ”AP finished for AP_ID <NNNN>”
”Stopped execution of AP: <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>,
PC : <NNNN>. HALT completion code: <SUCCESS/FAILURE> & <NNNN>”
INFO UCLI ”AP finished for AP_ID <NNNN>”
”Stopped execution of AP: <PATHNAME> at line <NNNN> : stopped inside library <PATHNAME> at line <NNNN>, Instruction : <INSTRUCT>, PC : <NNNN>. HALT completion code: <SUCCESS/FAILURE> & <NNNN>”
INFO UCLI ”AP suspended for AP_ID <NNNN>”
”<PATHNAME> suspended.”
INFO UCLI ”AP resumed for AP_ID <NNNN>”
”<PATHNAME> resumed.”
INFO UCLI ”AP, ID <NNNN>, terminated via the TES RPI.”
”AP <PATHNAME> terminated on request from <NAME>.”

TYPE GROUP Text

MSG UEVT ”<AP_TEXT>”
”<AP_TEXT>”

Copyright per DIN 34

D-3.3.3 Messages from DBS (Test Result DB)

Comms error in DBS_COMMS.SEND_TO_FA_WITH_ACKNOWLEDGE
SEND_MESSAGE returns NW_BADDSTID
—> Is the FA SAS installed ?

Comms error in DBS_COMMS.SEND_TO_FA_WITH_ACKNOWLEDGE
SEND_MESSAGE returns ETIMEDOUT
—> Is the FA SAS running ?

END_ERROR raised during LOCAL_PRINT_SERVICES.CHECK_PRINT

ERROR in CENTRAL_EXEC_DISPATCHER.MAIN_TASK.",
MESSAGE TYPE = INIT_EXECUTION_SESSION
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = NOT_ENOUGH_PACKED_DATA. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CENTRAL_EXEC_DISPATCHER.MAIN_TASK
MESSAGE TYPE = STORE_EVENT_SESSION_FILE",
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = FATAL_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CEN_FILE_SERV.STORE_FILE
File size: 50750464, Free space: 46821376
E_FILE_SYSTEM_FULL. Cause: Not enough space. Consequence: STORE FILE fails. Recovery act: See DBS_ERR_102. Debug info: STORE_FILE in dbs_central_files_services.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
ADT_PACKED_FILE fails for DELETE on /testnode/cgs–test1/HCI_01.03–03–98_15:30:09",
"E_STORE_EXECUTION_RESULT_FILE_16. Cause: ADT_PACKED_FILE error. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
Session PATRICK, File /testnode/cgs–test1/HCI_01.03–03–98_15:30:09:024.EVT.034
E_STORE_EXECUTION_RESULT_FILE_11. Cause: Cannot store EVENT file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_312. Debug info: STORE_EVENT_FILE in dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.",
ADT_PACKED_FILE fails for DELETE on /testnode/cgs–test1/HCI_01.03–03–98_15:30:13
E_STORE_EXECUTION_RESULT_FILE_16. Cause: ADT_PACKED_FILE error. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_central_log_data_management.a

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE
"Session PATRICK, File /testnode/cgs–test1/HCI_01.03–03–98_15:30:13:120.EVT.035",
"E_STORE_EXECUTION_RESULT_FILE_11. Cause: Cannot store EVENT file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_312. Debug info: STORE_EVENT_FILE in dbs_central_log_data_management.a

Copyright per DIN 34

ERROR in CEN_LOG_DATA_MGMT.STORE_EVL_FILE.
APPEND_LOCAL_EVL_TO_CENTRAL returns DBS_INTERNAL_PROBLEM, Producer TES_01, Session
E_STORE_EXECUTION_RESULT_FILE_07. Cause: Cannot store EVL file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_304. Debug info: STORE_EVL_FILE in dbs_central_log_data_management.a

ERROR in CEN_TAB_MANA INT_INSERT_EVT_FILE_INTO_TAB
E_INTERNAL_INSERT_EVENT_FILE_INTO_TABLE_01. Cause: Wrong file format (/testnode/csf_hp2/TES_01.24–02–98_15:27:00:768.EVT.165). Consequence: Current operation fails. Recovery act: See DBS_ERR_208. Debug info: INTERNAL_INSERT_EVENT_FILE_INTO_TABLE in dbs_ce

ERROR in CLDM.STO_EVL_FL.APPEND_LOC_TO_CEN.
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = constraint_error. Consequence: Current operation fails. Recovery act: None. Debug info: None.

ERROR in CLDM.STO_EVL_FL.APPEND_LOC_TO_CEN
Producer TES_01, Session UWE, File /testnode/csf_hp2/TES_01.24–02–98_15:26:57:97
E_STORE_EXECUTION_RESULT_FILE_09. Cause: Cannot store EVL file. Consequence: Current operation fails fails. Recovery act: See DBS_ERR_302. Debug info: STORE_EVL_FILE in dbs_central_log_data_management.a

ERROR in DB_BASIC_SERVICES.CONVERT_CONFIGURATION_FILE_KIND
Cannot convert to DATA_CATALOG_TYPES.T_CONFIGURATION_FILE_KIND
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = CONVERSION_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_db_basic_services.pad

ERROR in DB_BASIC_SERVICES.ROLLBACK.
”SQL_ERROR raised –1041, ORA–01041: internal error. hostdef extension doesn’t exi”,
”E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_db_basic_services.pad

ERROR in GET_FILE_SIZE
”UNIX_CALLS.STAT returns ENOENT for /testnode/sivq–tn1//archive/TES_01132_1998080”,
”E_DBS_UNIX_SERVICES_01. Cause: Cannot get file size. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_unix_services.a

ERROR in INIT_LOCAL
”CONNECT_TO_VICOS_PRODUCT returns ECONNREFUSED”,
”E_INIT_LOCAL_02. Cause: Cannot connect to CENTRAL_EXEC_PRODUCT. Consequence: The application is not connected to Central DBS. Recovery act: None. Debug info: INIT_LOCAL in dbs_central_comms.a

ERROR in MASTER_ARCHIVE.GET_MA_CONFIG_REFS.
E_OTHERS_OF_EXCEPTION. Cause: Exception OTHERS raised. EXCEPTION_NAME = CONVERSION_ERROR. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_file.pad

Copyright per DIN 34

ERROR in MA_EXECUTION.GET_EXEC_NAMES_SEL_CRIT_INT
SQL_ERROR raised –3114, ORA–03114: not connected to ORACLE
E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_execution.pad

ERROR in MASTER_ARCHIVE.ADD_MA_EVENT_FILE_REF
SQL_ERROR raised –3114, ORA–03114: not connected to ORACLE
E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_ma_logged_data_file.pad

ERROR in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE
ADD_MA_EVENT_FILE_REF returns DBS_ORACLE_PROBLEM, Session DEFAULT_TEST_SESSION,E_STORE_EVENT_FILE. Cause: Adding event file ref fails. Consequence: Current operation fails. Recovery act: See DBS_ERR_311. Debug info: Prcedure STORE_EVENT_FILE of dbs_central_log_data_management.a

ERROR in ONLINE_REFERENCE.GET_NUMBER_OF_EVALUATION_USER.",
SQL_ERROR raised –1092, ORA–01092: ORACLE instance terminated. Disconnection for
E_ORACLE_PBM. Cause: Oracle problem. Consequence: Current operation fails. Recovery act: None. Debug info: dbs_online_reference.pad

NAME_ERROR raised during L_SW_VAR_A.GET_TRDB_INFO
(no supplement)

(every minute) TES_01 TRDB SEVERE DBS_INT_PROBL MSG_# 13
—> when HCI is running: Combination with msg:
Could not read HK–values from DBS_RPI
DBS HK–values will not be written. Received return status
DBS_UNIX_PROBLEM from DBS_RPI.GET_TRDB_INFO.
SW Unit: $RCSfile: p_dbs_hk_collector_.a,v $
—> The connection to Central DBS might be disturbed
—> Check if a file $DBS_HOME/data/adatmp<tmp> is created recently (minutes ago)
—> if not : raise SPR

ORACLE_INIT_FAILED in FA_LISTENER.MAIN_TASK

—>The central DBS processes had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

ORACLE_INIT_FAILED in CENTRAL_EVAL_DISPATCHER.RUN_EVAL

—>The central DBS process serving the evaluation user had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

ORACLE_INIT_FAILED in CENTRAL_EXEC_DISPATCHER.MAIN_TASK

—>The central DBS process serving the online test execution had a problem to connect to Oracle
—> Is Oracle running ? Search for additional messages from DBS

STATUS_ERROR raised during L_SW_VAR_A.GET_TRDB_INFO

—> ???

Copyright per DIN 34

STORAGE_ERROR raised during LOCAL_PRINT_SERVICES.CHECK_PRINT

—> went out of memory
—> see below for workarounds on problems

–SQL_ERROR in DB_BASIC_SERVICES.CONNECT_TO_ORACLE

ORA–01034: ORACLE not available

—> The Oracle Processes are not running
—> Call your system administrator
—> The jprocesses are running on the DB Server node. They should be
—> available after boot of the OS.
—> They can also be started using the SQLDBA tool of Oracle.

SQL_ERROR in ONLINE_REFERENCE.GET_NUMBER_OF_EVALUATION_USER

ORA–03113: end–of–file on communication channel

SQL_ERROR in MA_EXECUTION.GET_EXEC_NAMES_SEL_CRIT_INT
ORA–03114: not connected to ORACLE

—> Are the Oracle Processes Running ?
—> Have the Oracle Prcesses been restarted when DBS was running
—> (i.e. are they started later than DBS ?)
—> go to Oracle / DB Server and verify via ps
—> Restart Oracle and/or DBS processes
—> Restart your application
—> (Note: TSCV will restart DBS automatically during startup)

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.

ORA–00942: table or view does not exist

INSERT_EVENTS_INTO_TABLE_PROBLEM in CEN_LOG_DATA_MGMT.STORE_EVENT_FILE.
ERR_311: Local EVENT not stored to Central.DEFAULT_TEST_SESSION,
/GSAF_HOME/dbs/data/WORK/DEFAULT_TEST_SESSION/TSCV....EVT.810>,
DBS_ORACLE_PROBLEM
—> ?? (user access problem ?? SPR ??)

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.

ORA–01653: unable to extend table space

—> The tablespace EVENT_SPACE could not be extended
—> Verify via sqlplus:
SQL > select * from user_free_space where tablespace_name = ’EVENT_SPACE’;
—> Remove /archive old test sessions
—> Close actual test session by DBS recovery scripts**,** if still open
—> Restart DBS on DB Server

SQL_ERROR in CEN_TAB_MANAG INSERT_EVT_BUFF_INTO_TABLE.

ORA–01858: a non–numeric character was found where a numeric was expected

—> Something went wrong when inserting an event into the TRDB log
—> Write SPR

Copyright per DIN 34

SQL_ERROR in DB_BASIC_SERVICES.CONNECT_TO_ORACLE
ORA–01034: ORACLE not available

—>The application tried to connect to DBS, which in turn tried to connect to Oracle. But Oracle
—> is not available
—> Check on DB_Server node with ps –auxw | grep ora
—> are all processes running: ora_pmon_<oraclesid>
—> ora_dbwr_<oraclesid>
—> ora_lgwr_<oraclesid>
—> ora_smon_<oraclesid>
—> orasrv
—> Startup these processes and restart the application

***** WARNING from CENTRAL_EXEC_DISPATCHER.MAIN_TASK**.
PROCESS_STOP_REQUEST returns DBS_COMMUNICATION_PROBLEM
W_STOP_CENTRAL_03. Cause: Stop Request Refused. Consequence: Central DBS is not stopped. Recovery act: None. Debug info: PROCESS_STOP_REQUEST in dbs_central_exec_dispatcher.a

***** WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER**
Error DBS_INTERNAL_PROBLEM occurred when Cyclic_Processing
—> Is Central DBS still running an OK ?

***** WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER**
Error DBS_UNIX_PROBLEM occurred when Cyclic_Processing
—> There might be a problem accessing files / directories / disk
—> under $DBS_HOME/data/WORK

***** WARNING from L_CY_EVENT_SEND.CYCLIC_EVENT_SENDER.**
”CYCLIC_EVENT_PROCESSING returns DBS_UNIX_PROBLEM”,
”W_CYCLIC_EVENT_PROCESSING. Cause: Internal error. Consequence: Events are possibly lost. Recovery act: None. Debug info: dbs_local_cyclic_event_sender.a

Copyright per DIN 34

Messages from Central DBS:

DBS_ERR_101 : The asynchronous request to remove Evaluation Result file failed :
Followed by : ”<’Filename’>.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS failed to complete a deletion request. ”
” The Unix Error Number was reported in a preceding Error message. ”
RECOVERY_END:

DBS_ERR_102 : The asynchronous request to store a file into the TRDB failed because the TRDB disk is full:
RECOVERY_START:
” Central DBS failed to complete store file request, Disk is full. ”
” – If the file concerned was a Raw Data file or an Execution ”
” Result file, a synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency :”
” file is left at initial location without any reference in TRDB.”
” – If the file concerned was not a Raw Data file or an Execution ”
” Result file, a reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” This applies for Evaluation Session files.”
” The file must be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_103 : The request to store a Raw Data file into the TRDB failed during addition of Oracle references, an exception was raised:
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Storage of file into TRDB failed, Central DBS failed to add Raw Data”
” file references into ORACLE TRDB.”
” A synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency:”
” file is left at initial location without any reference in TRDB.”
RECOVERY_END:

DBS_ERR_104 : The request to store an Execution Result file into the TRDB failed during addition of Oracle references, an exception was raised:
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Storage of file into TRDB failed, Central DBS failed to add Execution Result”
” file references into ORACLE TRDB.”
” A synchronous Status has been given back to the Requester, ”
” there is no reference to the file to store and no Inconsistency:”
” file is left at initial location without any reference in TRDB.”
RECOVERY_END:

Copyright per DIN 34

DBS_ERR_105 : Central DBS failed to return the acknowledge (following) to Local node during storage of Raw Data file or Exec Result file into the TRDB :
Preceeded by : 'failure status'
Followed by : "'ACK to provide' to 'Producer' : <'Filename'>, 'Session'."
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" During storage of an Execution Result file or a Raw Data file,"
" Central DBS failed to return the status of the operation to the"
" requesting application."
" This message could explain a DBS_COMMUNICATION_PROBLEM or
" COMMS_TIME_OUT status"
" obtained on a local node."
RECOVERY_END:

DBS_ERR_106 : Central DBS failed to update file refs during asynchronous storage of file into the TRDB :
Followed by : "<'Filename'>, 'Session', 'Producer', 'File type'."
RECOVERY_START:
" Storage of a file into TRDB failed: Central DBS failed to update "
" file references with the new location of the file."
" The file is not deleted from its initial location, and its ORACLE"
" references are pointing to initial location. The file could be "
" already copied into its final destination (under Central TRDB). "
" The file should be deleted using the Recovery Scripts "
" <File Storage Failure> Menu."
RECOVERY_END:

DBS_ERR_107 : Central DBS failed to delete the file just stored into the TRDB from its initial location. It is stored into TRDB but stays on its initial location.
Followed by : "<'Filename'>, 'Session', 'Producer', 'File type'."
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" File stored into TRDB but Central DBS failed to delete it "
" from its initial location."
" This problem does not lead to any inconsistency. The file"
" specified in the message should however be deleted manually."
RECOVERY_END:

DBS_ERR_108 : Central DBS failed to copy file into TRDB directory structure.
Followed by : "<'Filename'>, 'Session', 'Producer', 'File type'."
RECOVERY_START:
" Store of file failed, Central DBS failed to copy/move the"
" file from its initial location into TRDB. The failure"
" reasons are detailed in a preceeding message (e.g. file name"
" given is not valid, read/write permissions not set correctly...)"
" An ORACLE reference has been created for the file to store, "
" file is left at initial location and is referenced into TRDB."
" This applies for Evaluation Session files."

Copyright per DIN 34

” The file should be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_109 : The file type (result type) given for a file to store into TRDB
is not consistent (e.g. Exec Result file with result type set to DATA SET,
non initialised file type...).
Followed by : ”<’Filename’>, ’Session’, ’Producer’.”
RECOVERY_START:
” Storage of Result file into TRDB failed; the Result type given is invalid ”
” (e.g. Execution Result file with result type set to DATA SET,...)
” An ORACLE reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” The file should be deleted using the Recovery Scripts ”
” <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_110 : Central DBS failed to create Application subdirectory to store the
Raw Data files created by that Application.
Followed by : ”<’subdirectory name’>, <’Raw Data file name’>, ’Session’.”
RECOVERY_START:
” Store of file failed, Central DBS failed to create ”
” the Application subdirectory to store the Raw Data files”
” into TRDB. The failure reasons are detailed in a preceeding ”
” message (e.g. directory structure is corrupted...)”
” An ORACLE reference has been created for the file to store. ”
” file is left at initial location and is referenced into TRDB.”
” The file reference should be deleted using the Recovery Scripts”
” <File Storage Failure> Menu, and the TRDB file structure should be checked.”
RECOVERY_END:

DBS_ERR_111 : Central DBS failed to access a UNIX file/directory or an ORACLE table
when building list for automatic archiving. Auto–archiving processing is aborted.
Followed by : ”<’object_name’> ’object_type’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving failed: Central DBS failed to collect ”
” necessary information to decide on which files had to be archived ”
” to save space. This message is following another one describing ”
” the problem encountered (e.g. Oracle down, directory not accessible...). ”
” This message alone does not give an indication of inconsistency.”
RECOVERY_END:

DBS_ERR_112 : Auto–archiving succeeded but did not free enough space on disk
to raise secure level.
Followed by : ”(’SECURE_SPACE’ %) on <’central home directory name’>”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”

Copyright per DIN 34

” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving saved all available data of the session whose ARCHIVE_MODUS”
” is set but failed to reach the SAFE level specified in the dbs configuration”
” file (the Automatic Archiving is starting when disk space used reaches a CRITICAL ”
” level, it archives files of the session whose ARCHIVE_MODUS is set”
” until a SAFE level is reached). ”
” That means either the CRITICAL and SAFE level are not defined in accordance ”
” to the local configuration or the disk space is used mainly by other sessions.”
” This message alone does not give an indication of inconsistency, it”
” shows a non–optimum configuration for the Automatic Archiving functionnality.”
RECOVERY_END:

DBS_ERR_113 : Central Archive failed to archive all remaining file of a session,
The problem desciption follows.
Session is not archived.
Followed by : ”’Session Name’, ’problem desciption’”
RECOVERY_START:
” Automatic Archiving failed: Central DBS failed to archive remaining files ”
” of the session when its closure was requested.”
” The session is not archived and its state is kept to the value To_Be_Archived.”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered.”
” Session must be recovered by the Recovery Scripts <Session is Used> Menu”
” but specific actions could be required according to preceeding messages.”
RECOVERY_END:

DBS_ERR_114 : ARCHIVING_ERROR :
Auto–archiving failed, Central Archive Failed to Archive the list
of selected files for a given reason (This message is a high level
one to warn the user of the consequences, it should follows others
describing the problem on Central Arch or on FA–SAS).
Followed by : ”<’Session Name’>, ’Status of problem’”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Automatic Archiving failed to save available data of the session whose ”
” ARCHIVE_MODUS is set.”
” This message follows others giving a precise description of the ”
” problem encountered.”
RECOVERY_END:

DBS_ERR_115 : INCONSISTENCY : The deletion of a file just archived
on FA for a given Session failed. This file is still
on the TRDB disk but its references are updated (ARCH + path)
The Session Record is not updated (To Be ARchived) and the
Session directories must be deleted.
There is an inconsistency in the TRDB.
Followed by : ”<’file Name’>”
RECOVERY_START:
” The deletion of a file just archived on FA medium for the given session”

Copyright per DIN 34

” failed. File is archived on FA medium and still on the TRDB disk. It is”
” referenced as archived (on FA medium). The Session info is not updated”
” and is kept in the state To_Be_Archived.”
” The file has to be deleted from the TRDB disk and the Session must be”
” recovered using the Recovery Scripts <Session is Used> Menu.”
” This message is preceeded by another one giving a precise description”
” of the problem encountered (e.g. permission denied, NFS stale...).”
RECOVERY_END:

DBS_ERR_116 : INCONSISTENCY : The updating of references of a file just archived
on FA for a given Session failed. File is archived on FA medium
but its new location is not referenced in Oracle database
There is an inconsistency in the TRDB.
Followed by : ”<’fileName’>”
RECOVERY_START:
” The update of the reference for a file just archived on FA medium”
” failed. This message is preceeded by another one giving a precise description”
” of the problem encountered (e.g. Oracle down...).”
” A list of files has been archived on the FA medium; they are still on”
” the TRDB disk and referenced as so. The Session info is not updated”
” and is kept in the state To_Be_Archived. The Session must be”
” recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_117 : INCONSISTENCY : The operation ARCHIVE_SESSION_ON_FA failed.
The update of a session status failed after this session data
has been archived on FA medium, the data references updated and
the data deleted from TRDB disk.
There is an inconsistency in the TRDB.
Followed by : ”’sessionName’”
RECOVERY_START:
” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA but it failed”
” to update the session status. This message could be preceeded by ”
” another one giving a precise description of the problem encountered ”
” (e.g. Oracle down...).”
” The Session info is not updated and is kept in the state To_Be_Archived.”
” The on–line data is also left to an out of date state.”
” The Session data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_118 : INCONSISTENCY : The operation ARCHIVE_SESSION_ON_FA failed.
DBS successfuly archived the files on the FA but it failed
to remove the directories of the session on the TRDB disk.
The File references are updated (ARC + new path).
The Session Record is not updated (To Be ARchived) and the
Session directories must be deleted.
There is an inconsistency in the TRDB.
Followed by : ”’Session Name’”
RECOVERY_START:

Copyright per DIN 34

” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA,”
” updated the file references (archived) but it failed”
” to delete the Session data from the TRDB disk. This message is”
” preceeded by another one giving a precise description of the problem”
” encountered (e.g. Permission denied, NFS stale...).”
” The Session data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_121 : INCONSISTENCY : The update of online data table
failed after this session data is archived on FA medium and session state modified.
There is an inconsistency in the TRDB.
Followed by : ”’sessionName’”
RECOVERY_START:
” The operation ARCHIVE_SESSION_ON_FA failed. ”
” Central DBS successfuly archived the session files on the FA, ”
” updated the session status but it failed to update on–line data references.”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered (e.g. Oracle down...).”
” The on–line data must be recovered using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_201 : INCONSISTENCY : The creation of a new Oracle Event Table for a given
Test Execution Session failed because this table already
existed. There is an inconsistency in the TRDB.
Followed by : ”<’Session Name’> ”
RECOVERY_START:
” The operation INITIALIZE_EVENT_TABLE failed. ”
” Central DBS detected that the event table it was supposed to create”
” already existed. There is only one event table per session and”
” that problem should not happen. It shows an inconsistency that can be ”
” related to other problems, e.g. deletion failure, invalid Oracle data base.”
” The event table should be deleted by the Recovery Scripts if no other”
” action is deduced from other inconsistencies (<Delete Session> Menu).”
RECOVERY_END:

DBS_ERR_202 : The Import of an Oracle Raw Data File Reference Table for a given
Test Execution Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of an Oracle Raw Data File Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

Copyright per DIN 34

DBS_ERR_203 : The Import of an Oracle Config File Reference Table for a given
Test Execution Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of an Oracle Configuration File Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_204 : The Import of an Oracle Session Reference Table for a given
Test Session failed because the file where this table has
been exported has a wrong format..
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Session Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_205 : The deletion of an event table failed with Time out. Central DBS failed
to lock the table to remove it. After a given number of trial, it
interrupted the operation.
Followed by : ”’Session Name’, ’table name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Session Reference Table for the given”
” Test Execution Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_206 : The operation MANAGE_AA_SESSION_EVENT_SPACE failed : this operation
aims to move the content of an Oracle Event Table into a file referenced in the TRDB and
to archive it. The Table content has been copied into a given file and the Table references
have been updated with this file name, but then, the deletion of the event
table failed. The table must be deleted by the recovery scripts
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:

Copyright per DIN 34

” The deletion of an event table failed with Time out. Central DBS failed”
” to lock the table to remove it. After a given number of trial, it”
” interrupted the operation. It means that an Evaluator user is continuously”
” using the table. Check that no Local Node is blocked in an Evaluation operation”
” concerning events of that session.”
” This message is probably accompanied by others describing the higher level”
” implications of the problem; these could imply corrective actions.”
RECOVERY_END:

DBS_ERR_209 : The Import of an Oracle Result File Reference Table for a given
Session failed because the file where this table has been exported
has a wrong format.
Followed by : ”<’File Name’, ’Session Name’> ”
RECOVERY_START:
” The Import of a Session from FA medium failed.
” The Import of the Oracle Table containing Result file references”
” for the given Session failed because the file where this table has”
” been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_213 : Import of a session from the Final Archive failed.
The file referenced as Export file for the Session Engineering Value File References
has not the correct format.
Followed by : ”’File name’, ’New Session Name’”
RECOVERY_START:
” The Import of a Test Execution Session from FA medium failed.
” The Import of the Oracle Table containing Eng. Value file references”
” for the given Test Execution Session failed because the ”
” file where this table has been exported has a wrong format.”
” Data and references are left on the TRDB disk.”
” The imported session (NEW_NAME) should be deleted by the ”
” Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_215 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to create a new Event Oracle table for the
session after it archived the current one to save space. The status of
the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has archived the current Event Table to save space and”
” then has not been able to create a new empty Event Oracle table for”
” the given Session. The failure status is part of the message.”
” Other messages are related.”

Copyright per DIN 34

” The session should be deleted using the recovery scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_216 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to archive the current Event Oracle table
to save space. The status of the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. The failure status is part of the message.”
” Other messages are related.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_217 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to archive the current Event Oracle table
to save space.
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. An unexpected problem occured.”
” The failure status is part of the message.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_218 : Management of Event space for Automatic Archiving failed.
Central DBS has not been able to get the Event Space free space to
check if Automatic archiving of Session Event table is required.
The status of the operation is part of the message.
Preceeded by : ’failure status’,
Followed by : ”’Session Name’”
RECOVERY_START:
” The Management of Event space for Automatic Archiving failed.
” Central DBS detected a need for the Automatic archiving of the Events”
” of the session. It has not been able to archive the current Event Oracle table
” to save space. The failure status is part of the message.”
” Other messages are related.”
” This message could have important consequences on the session data integrity if”
” the Event ORACLE Table Space is full and no action is taken to allow archiving.”
RECOVERY_END:

DBS_ERR_219 : INCONSISTENCY : The deletion of Default Session Events brougth an
inconsistency into the TRDB: after having deleted the Events selected,
Central DBS failed to update the reference of the Event table (time frame refs).
Preceeded by : ’failure status’,

Copyright per DIN 34

Followed by : "'new begin date'"
RECOVERY_START:
" The deletion of Default Session Events brougth an inconsistency into the TRDB:
" after having deleted the Events selected, Central DBS failed to update the "
" reference of the Event table (time frame LT and SMT)."
" The DEFAULT_TEST_SESSION must be recovered using the <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_220 : Retrieve of Events from FA: Events were retrieved from the Final archive into an Oracle Table; the references concerned were updated but Central DBS failed to remove an Event file used for the retrieving. A warning showing the file name is issued to allow the user to delete it manually.
Followed by : "'file name', 'session name'"
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" Events were retrieved from the Final archive into an Oracle Table. "
" The references concerned were updated but Central DBS failed to remove the temporary"
" Event file used for the retrieving."
" This problem does not lead to any inconsistency. The file"
" specified in the message should however be deleted manually."
RECOVERY_END:

DBS_ERR_221 : The operation STORE_EVENT_TABLE failed : this operation aims to move the content of an Oracle Event Table into a file referenced in the TRDB. The Table content has been copied into a given temporary file and the Table references have been updated with this file name, but then, the storage of this temporary file into TRDB failed. The Event table is not deleted.
Preceeded by : 'failure status',
Followed by : "'Session Name', 'file name'"
RECOVERY_START:
" Storage of a file into TRDB failed. The operation STORE_EVENT_TABLE failed."
" This operation aims to move the content of an Oracle Event Table into a "
" file referenced in the TRDB. The Table content has been copied into a given "
" temporary file and the Table references have been updated with this file name."
" After that, the storage of this temporary file into TRDB failed."
" The Event table is not deleted."
" The file should be deleted and the Oracle table references updated,"
" using the DBS Recovery Scripts <File Storage Failure> Menu."
RECOVERY_END:

DBS_ERR_301 : Creation of a new central EVL file for local EVL file storage failed. A file and its references are created but not usable as central EVL file. The file may be unmoved to TRDB, or its references are not updated after being moved to TRDB.
Followed by : "<file name without path>"
RECOVERY_START:
" Storage of an Eng. Value file into the TRDB failed."
" The temporary file should be deleted and the Oracle table references updated, "
" using the DBS Recovery Scripts <File Storage Failure> Menu."
RECOVERY_END:

Copyright per DIN 34

DBS_ERR_302 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Only a part of the Local Eng. Value File
is stored into the TRDB.
Followed by : "'Session Name', 'Local Node', 'Local Eng. Value File Name', 'Central Eng. Value File Name'"
RECOVERY_START:
" Storage of an Eng. Value file into the TRDB failed. Only a part of the Local Eng. Value "
" file is stored into the TRDB."
" The temporary file should be deleted and the Oracle table references updated, "
" using the DBS Recovery Scripts <File Storage Failure> Menu."
RECOVERY_END:

DBS_ERR_303 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. No data is stored into TRDB.
Followed by : "'Session Name', 'Local Node', 'Local Eng. Value File Name'"
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" Storage of an Eng. Value file into the TRDB failed. No data from the Local Eng. Value "
" file is stored into the TRDB."
" No reference for the local file has been created."
RECOVERY_END:

DBS_ERR_304 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Central DBS has been able to create
an Oracle reference for that Local Eng. Value file.
Followed by : "'Session Name', 'Local Node', 'Local Eng. Value File Name'"
RECOVERY_START:
" Storage of an Eng. Value file into the TRDB failed. "
" Central DBS has been able to create an Oracle reference for that Local Eng. Value file."
" The temporary file should be deleted and the Oracle table references updated, "
" using the DBS Recovery Scripts <File Storage Failure> Menu."
RECOVERY_END:

DBS_ERR_305 : Storage of an Eng. Value File from a Local Node into
the TRDB failed. Central DBS has been able store the Eng. Values
into TRDB but failed to update the Oracle references.
Followed by : "'Central Eng. Value File Name', 'Session Name', 'Local Node'"
RECOVERY_START:
" Storage of an EVL file into the TRDB failed. "
" Central DBS has been able to create an Oracle reference for that Local Eng. Value file."
" It failed to update the Oracle reference."
" The temporary file should be deleted and the Oracle table references updated, "
" using the DBS Recovery Scripts <File Storage Failure> Menu."
RECOVERY_END:

DBS_ERR_306 : Storage of an Eng. Value File from a Local Node into
the TRDB successful but Central DBS failed to delete the Local file.
Followed by : "'Session Name', 'Local Node', 'Local Eng. Value File Name'"
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."

Copyright per DIN 34

” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to delete the Local file from its initial location.”
” This problem does not lead to any inconsistency. The file”
” specified in the message should however be deleted manually.”
RECOVERY_END:

DBS_ERR_307 :
RECOVERY_START:
” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to delete the Local file references in Oracle.”
” The Oracle table references must be updated using the DBS Recovery ”
” Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_308 :EVL stored to Central, but Central EVL file size check failed. May be too big, and still with ’CNA’ status. ”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
”This message is following others describing the problem with more details”
” Storage of an EVL file into the TRDB is successful (data and Oracle references), ”
” but Central DBS failed to check the size of the Central EVL file appended or”
” to update its reference to Central Accessible.”
” This message shows a general problem that could have future consequences”
” (e.g. another Oracle update failure or another problem accessing unix flat files)
” This problem could lead to others and the related messages should be analysed.”
RECOVERY_END:

DBS_ERR_309 : EVL stored to Central, but auto–archiving check failed. May be not enough disc space.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” Automatic Archiving failed: Central DBS has not been able to check the ”
” available space or to process with the Automatic Archiving if required. ”
” This message could be preceeded by another one giving a precise description”
” of the problem encountered (Comms problem, Unix path not accessible...).”
” Consequently, the free space on the TRDB disk could be under the”
” expected limit. This problem could lead to others and the related messages ”
” should be analysed. Specific actions could be required according to”
” preceeding messages.”
RECOVERY_END:

DBS_ERR_311 : Local Event file not stored into TRDB
RECOVERY_START:
” Storage of an Event file into the TRDB failed. ”
” Central DBS has not been able to create an Oracle reference for this Event file or”
” it failed to insert the file content into the Oracle event table.”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_312 : EVENTs from local file stored into TRDB, but local file not deleted
RECOVERY_START:

Copyright per DIN 34

” Storage of an Event file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for this Event file and”
” to insert the file content into the Oracle event table. But it failed to delete”
” the file. The local file and its reference remain.
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_313 : EVENTs from local file stored into TRDB, local file deleted, but not its Oracle reference
RECOVERY_START:
” Storage of an Event file into the TRDB failed. ”
” Central DBS has been able to create an Oracle reference for that Event file and”
” to insert the file content into the Oracle event table. It has deleted”
” the file but failed to remove its reference.
” The temporary Oracle table references must be removed, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_314 :EVENTs from local file not stored into TRDB due to unexpected problem
RECOVERY_START:
” Storage of an Event file into the TRDB failed due to an unexpected problem. ”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_315 : Central failed to supply ’GET_EVL_LIST’ Acknowledge
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem trying to send acknowledge of the given”
” operation to the caller. The problem could be described in a preceeding message.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT status”
” given on a local node.”
RECOVERY_END:

DBS_ERR_316 : Eng. Values from local file not stored into TRDB due to unexpected problem
RECOVERY_START:
” Storage of an Eng. Value file into the TRDB failed due to an unexpected problem. ”
” The temporary file should be deleted and the Oracle table references updated, ”
” using the DBS Recovery Scripts <File Storage Failure> Menu.”
RECOVERY_END:

DBS_ERR_317 : During Session closure, DBS failed to store a file whose name is not formatted as an EVL or EVENT file”
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem while closing a session (session name is given)”
” The WORK directory for that session contains a file whose name is not formatted”
” as an EVL or EVENT file. This file is not expected at that location.”
RECOVERY_END:

Copyright per DIN 34

DBS_ERR_401 : EVENT_TABLE_NOT_REMOVED
During initialisation of execution session, the Commit
failed with a fatal error.
Participating Applications Flag and Session dir space
(including stored Config files) has been removed; The FA
Auto Arch reservation are released.
Event Table can't be removed after the fatal error.
Followed by : "'execution session name'".
RECOVERY_START:
" During initialisation of execution session, ORACLE failed to commit"
" the new session references."
" DBS tried to recover partially by :"
" – resetting the Participating Applications Flags (to warn Participating Applications"
" of a session opening/closure), "
" – removing the session directory structure (including stored Config files), "
" – releasing the FA SAS device reservation (in case of AUTO ARCH selected). "
" The Event Table created for the new session cannot be deleted after ORACLE"
" generated a fatal error."
" The new session Event Table should be deleted by the Recovery Scripts as"
" soon as the ORACLE data base is in good state. <Delete Session> Menu"
RECOVERY_END:

DBS_ERR_403 : FILE_DELETION_FAILURE
During deletion procedure of default EVL, RD or RESULT files,
a problem occured while deleting one file.
Followed by : "'file name'".
RECOVERY_START:
" The operation Delete Default Session failed. "
" A problem described in a preceeding message occured while deleting a"
" DEFAULT_TEST_SESSION file. "
" The DEFAULT_TEST_SESSION is kept in the state To_Be_Partially_Deleted,"
" some references could be out of date. It should be recovered "
" using the Recovery Scripts <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_404 : DIR_DELETION_FAILURE (warning)
During execution session export, the temporary export
directory created by operation can't be deleted.
Followed by : "'dir path'".
RECOVERY_START:
" Central DBS exported successfully an Execution session"
" onto an FA medium; but it failed to Remove the temporary Export "
" directory structure created for that purpose."
" The temporary Export directory must be deleted using the Recovery Scripts"
" <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_405 : SESSION_DIR_SPACE_NOT_REMOVED (warning)
During execution session deletion, the session info are
updated on ORACLE but session dir space is not removed.

Copyright per DIN 34

Followed by : "'session name'".
RECOVERY_START:
" Central DBS successfully updated the ORACLE references"
" for a DELETE_EXECUTION_SESSION operation,"
" but it failed to Remove the session directory from TRDB disk."
" The session directory must be deleted using the Recovery Scripts <Delete Session> Menu"
RECOVERY_END:

DBS_ERR_406 : SESSION_DELETION_FAILURE (warning/error)
During execution session deletion, an exception occured.
The session is not deleted and may be in a inconsistent state.
Followed by : "'session name'".
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the DELETE_EXECUTION_SESSION operation."
" This message is preceeded by another one describing the problem."
" The session is left in an inconsistent state and should be recovered"
" using the Recovery Scripts <Delete Session> Menu."
RECOVERY_END:

DBS_ERR_407 : DATA_DELETION_FAILURE (warning/error)
During retrieved data deletion, an exception occured.
The data may be not deleted and session may be in an inconsistent state.
Followed by : "'session name', 'data to delete'".
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the DELETE_RETRIEVED_EXECUTION_DATA operation."
" This message is preceeded by another one describing the problem."
" The session is left in an inconsistent state and should be recovered"
" using the Recovery Scripts."
" The only way to recover is to delete all the on–line (retrieved) data"
" from the TRDB disk. The data can further be RETRIEVED from FA medium."
" Use the <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_408 : DEFAULT_SESSION_DELETION_FAILURE (warning/error)
During default session deletion, an exception occured.
The data may be not deleted and session may be in an inconsistent state.
Followed by : –/–.
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the DELETE_DEFAULT_TEST_SESSION operation."
" This message is preceeded by another one describing the problem."
" The session is left in an inconsistent state and should be recovered"
" using the Recovery Scripts <Delete Session> Menu."
RECOVERY_END:

DBS_ERR_409 : INVALID_SESSION_STATE
The operation failed : Current execution session state
is not the attended one.
Followed by : "'session name', <'invalid session state'>".

Copyright per DIN 34

RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Execution Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. ARCHIVE_EXECUTION_SESSION and DELETE_EXECUTION_SESSION) at the ”
” same time and that they both reset the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_410 : ARCHIVE_SESSION_FAILURE (warning/error)
During execution session archiving, an exception occured.
The session may be not archived and session may be in an inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the ARCHIVE_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_411 : RETRIEVE_SESSION_FAILURE (warning/error)
During execution session retrieving, an exception occured.
The data may be not retrieved and session may be in an inconsistent state.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the RETRIEVE_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_412 : EXPORT_SESSION_FAILURE (warning/error)
During execution session export, an exception occured.
The session is not successfully exported.
Followed by : ”’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_413 : IMPORT_SESSION_FAILURE (warning/error)
During execution session export, an exception occured.
The session is not successfully imported and may be in an inconsistent state.
Followed by : ”’session name’”.

Copyright per DIN 34

RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the IMPORT_EXECUTION_SESSION operation.”
” This message is preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_414 : EXPORT_PREPARATION_FAILURE (warning/error)
During execution session export, export of tables to files failed.
The session export is aborted.
Followed by : ”’error status’, ’session name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EXECUTION_SESSION operation. Export of tables to files failed.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_415 : TMP_DIR_DELETION_FAILURE (warning/error)
During execution session import, the temporary directory (EXPORT)
can’t be deleted after completion of table import operation.
The session import is aborted but imported files may be present on disc.
Followed by : ”’export dir path’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the IMPORT_EXECUTION_SESSION operation. The given temporary directory”
” cannot be deleted.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_416 : –/–
The closure of the Execution Session failed: Central
DBS failed to remove the directory structure warning the
local nodes of the session opening/closure.
The Execution Session is not closed.
Followed by : ”’Session name’, ’problem status’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the CLOSE_EXECUTION_SESSION operation. It failed to remove the directory structure”
” warning the participating applications of the session opening/closure.”
” The Execution Session is not closed.”
” This message can be preceeded by another one describing the problem.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Close Session> Menu.”
RECOVERY_END:

Copyright per DIN 34

DBS_ERR_417 : SUPPLY_ACK_FAILURE
The communication failed: Central DBS failed to send
acknowledge of the operation to the caller.
Followed by : "'operation acknowledge', 'communication error status'".
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" Central DBS encoutered a problem trying to send acknowledge of the given"
" operation to the caller. The problem could be described in a preceeding message."
" This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT
" status given on a local node."
RECOVERY_END:

DBS_ERR_418 : UPDATE_FAILURE_WITH_INCONSISTENCY
The retrieving of files from FA succeed.
The retrieveing of the Event Oracle table succeed.
Files and Event Oracle table are present
on TRDB but update of Oracle File or Session references failed.
Followed by : "'session name'".
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the RETRIEVE_EXECUTION_SESSION operation. The retrieving of files from FA succeeded."
" The retrieveing of the Event Oracle table succeeded."
" Files and Event Oracle table are present on TRDB but update of Oracle File or Session"
" references failed."
" The session is left in an inconsistent state and should be recovered"
" using the Recovery Scripts <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_419 : SESSION_STATE_FAILURE
During the export operation, the session state cannot be set back
to previous state given as argument. This is a severe error because
Session could be locked in a state where all action is forbidden.
Followed by : "'previous session state'".
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the EXPORT_EXECUTION_SESSION operation. The session state cannot be set back"
" to previous state given as argument."
" The session is left in an inconsistent state and should be recovered"
" using the Recovery Scripts <Session is Used> Menu."
RECOVERY_END:

DBS_ERR_420 : –/–
The closure of the Execution Session failed: Central
DBS failed to update EVL files Oracle refes.
The Execution Session is not closed.
Followed by : "'Session name', 'problem status'".
RECOVERY_START:
" Central DBS encoutered a problem during the execution of"
" the CLOSE_EXECUTION_SESSION operation. Central DBS failed update EVL"

Copyright per DIN 34

” files Oracle references.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_421 : SEND_WITHOUT_ACK_FAILURE
The closure of the Execution Session failed: Central Exec
failed to request Automatic Archiving of Session Remaining
files to Central Archive. The session is referenced as closed,
its Event table is stored into a file (refs commited)
The Execution Session is closed but not archive.
Followed by : ”’problem status’, ’Session Name’”.
RECOVERY_START:
” Automatic Archiving: Central DBS encoutered a problem during the execution of”
” the CLOSE_EXECUTION_SESSION operation. Central exec failed to send”
” a request to Central arch to archive the remaining data of the session.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Delete Session> Menu.”
RECOVERY_END:

DBS_ERR_422 : TMP_EVT_FILE_LEFT
During Archiving of an execution session, an error has been detected
The copy of session files on the FA_MEDIUM failed.
Central DBS tried to recover to the initial state of
the Session, it reset the Session State to Closed,
and tried to remove the temporary Event file containing
Session Events. This deletion failed File should be
deleted manually.
Followed by : ”’filename’, ’Session Name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the ARCHIVE_EXECUTION_SESSION operation. The copy of session files on”
” the FA_MEDIUM failed. Central DBS tried to recover to the initial state of”
” the session: it reset the Session State to Closed, tried to remove the temporary”
” Event file containing Session Events. This deletion failed.”
” The given file must be deleted manually.”
RECOVERY_END:

DBS_ERR_423 : REMOVE_EVENT_TABLE_FAILURE
The archiving of an execution session was successful
(refs updated, files moved) until the last operation.
The dropping of the event table failed, it should be
removed by the recovery scripts
Followed by : ”’Session Name’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the ARCHIVE_EXECUTION_SESSION operation. Central DBS failed to remove the Oracle”
” Event table after completion of the transfer to FA Medium.”
” The session is left in an inconsistent state and should be recovered”

Copyright per DIN 34

” using the Recovery Scripts <Delete Session> Menu.””
RECOVERY_END:

DBS_ERR_503 : The operation DELETE_EVALUATION_SESSION failed because the Session state is not set correctly. The Local DBS should have set it to the value ’To Be Partially Deleted’ or ’To Be Completely Deleted’. Then it is an Internal Problem: Followed by : ”’Session Name’, ’Session State’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. ARCHIVE and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_504 : Central DBS failed to return the acknowledge (following) to Local node during an operation on Evaluation session (Creation or Deletion):
Preceeded by : ’failure status’
Followed by : ”’ACK to return’, ’Session’.”
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS encoutered a problem trying to send acknowledge of the given”
” operation to the caller. The problem could be described in a preceeding message.”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT ”
” status given on a local node.”
RECOVERY_END:

DBS_ERR_505 : The operation ARCHIVE_EVALUATION_SESSION failed because the Session state is not set correctly. The Local DBS should have set it to the value ’To Be Archived’. Then it is an Internal Problem: Followed by : ”’Session Name’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
” No Recovery Scripts action has to be executed following that message.”
” Central DBS found an invalid value for an Evaluation Session State.”
” It means that Local Nodes called a particular combination of operations”
” (e.g. ARCHIVE and DELETE) at the same time and that they both reset”
” the Execution Session State without detecting the ”
” other request. This should not lead to any inconsistency. ”
” Note other messages related to the same session. ”
RECOVERY_END:

DBS_ERR_509 : The operation RETRIEVE_EVAL_SESSION_FROM_FA failed because the Session state is not set correctly. The Local DBS should have set it to the value ’To Be Imported’ as a New Name is given in

Copyright per DIN 34

arguments.
Then it is an Internal Problem:
Followed by : "'Session Name', 'Session State'".
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" Central DBS found an invalid value for an Evaluation Session State."
" It means that Local Nodes called a particular combination of operations"
" (e.g. RETRIEVE and DELETE) at the same time and that they both reset"
" the Execution Session State without detecting the "
" other request. This should not lead to any inconsistency. "
" Note other messages related to the same session. "
RECOVERY_END:

DBS_ERR_510 : The operation IMPORT_EVAL_SESSION_FROM_FA failed because the
Session state is not set correctly. The Local DBS should have
set it to the value 'To Be Imported'.
Then it is an Internal Problem:
Followed by : "'Session Name', 'Session State'".
RECOVERY_START:
" This error message alone does not describe an inconsistency in the TRDB."
" No Recovery Scripts action has to be executed following that message."
" Central DBS found an invalid value for an Evaluation Session State."
" It means that Local Nodes called a particular combination of operations"
" (e.g. IMPORT and DELETE) at the same time and that they both reset"
" the Execution Session State without detecting the "
" other request. This should not lead to any inconsistency. "
" Note other messages related to the same session. "
RECOVERY_END:

DBS_ERR_511 : DBS failed to remove the directory structure of an Evaluation
Session during the operation DELETE_EVALUATION_SESSION.
The Oracle References are updated (commited) but some (all)
Session files are still on the TRDB disk.
All the Session files on the TRDB disk must be deleted by the recovery
scripts.
Followed by : "'Status of failure', 'Session Name'".

RECOVERY_START:
" Central DBS failed to remove the directory structure of an Evaluation."
" Session during the operation DELETE_EVALUATION_SESSION."
" The Oracle References are updated (commited) but some (all)"
" Session files are still on the TRDB disk. "
" All the Session files on the TRDB disk must be deleted by the Recovery "
" Scripts <Delete Session> Menu. "
RECOVERY_END:

DBS_ERR_512 : SESSION_STATE_FAILURE
During the export operation, the session state cannot be set back
to previous state given as argument. This is a severe error because
Session could be locked in a state where all action is forbidden.

Copyright per DIN 34

Followed by : ”’previous session state’”.
RECOVERY_START:
” Central DBS encoutered a problem during the execution of”
” the EXPORT_EVALUATION_SESSION operation. The session state cannot be set back”
” to previous state given as argument.”
” The session is left in an inconsistent state and should be recovered”
” using the Recovery Scripts <Session is Used> Menu.”
RECOVERY_END:

DBS_ERR_801 : COMMS INIT FAILURE.
During a FA controller start, communication initialisation failed.
The FA controller is stopped.
Followed by : ”’channel number’ returned ’status’”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
” DBS Central Archive has not been able to start due to Communication”
” Services problems. Refer to the status given in the message.”
RECOVERY_END:

DBS_ERR_802 : ACK NOT SUPPLIED.
During a dispatcher operation, a supply acknowledge operation failed.
The caller has not received the expected operation status.
The COMMS return an error code.
Followed by : ”(’status’ for ’operation’ to ’caller’)”.
RECOVERY_START:
” This error message alone does not describe an inconsistency in the TRDB.”
”No Recovery Script action has to be executed following that message.”
” DBS Central Archive has not been able to supply an acknowledge to the caller ;”
” (local application) ”
” The Communication Services returned an error (part of the message).”
” This message could explain a DBS_COMMUNICATION_PROBLEM or COMMS_TIME_OUT ”
” status obtained on a local application.”
RECOVERY_END:

Copyright per DIN 34

D-3.4 Model Execution

D-3.4.1 HLCL on–line

When an error occurs in a command sequence during execution, the effect depends on the predefined TRAP variable. If *error trapping* is on, an error within a command interrupts the command sequence, as if the interrupt key combination had been pressed. Execution can then be resumed with the RESUME command. If the TRAP variable is set to false errors in the command sequence do not interrupt the execution of the sequence.

If *error trapping* is on the erroneous command is displayed in the ICP window and an explanatory text is displayed which indicates the type of error.

If HLCL is used as an interactive command language there are two types of errors which can occur. Figure 15 shows the way syntax errors are displayed while the user types a command on–line.

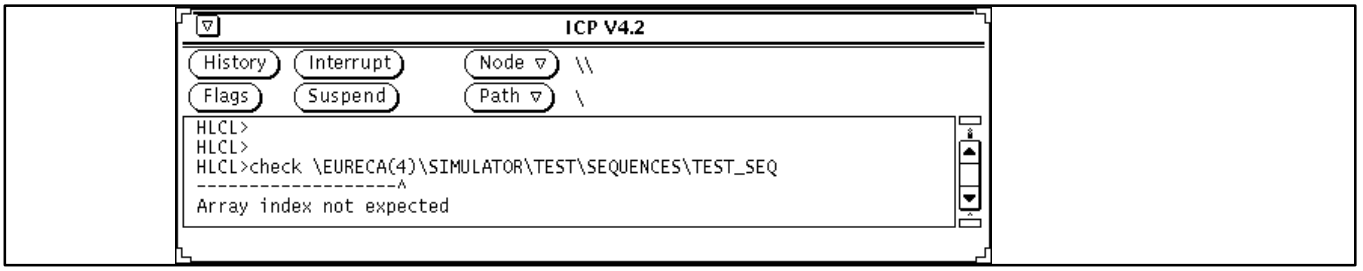


Figure 15 : *Syntax errors in the HLCL command are shown by a little arrow.*

Errors during command execution are displayed as shown in Figure 16. The error messages are written in plain and detailed text. The errors messages are self–explanatory, therefore a list of possible syntax errors is not provided.

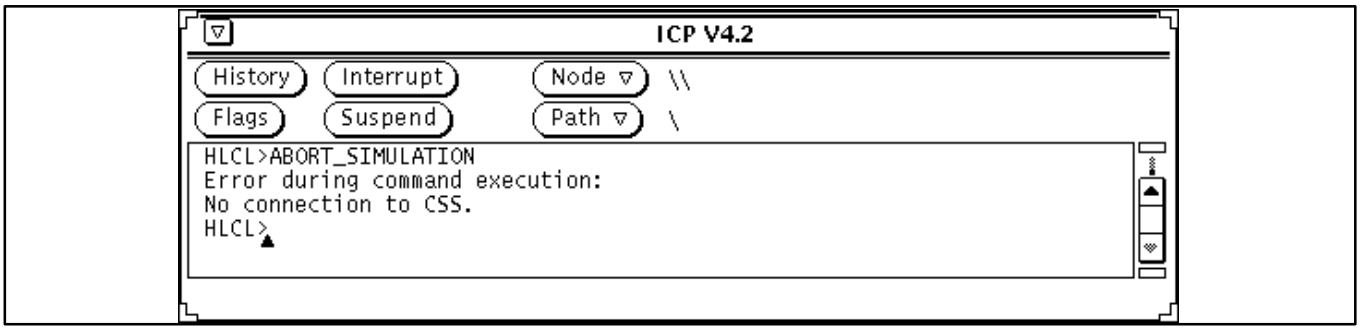


Figure 16 : *Errors during command execution*

D-3.4.2 Model Execution Messages

Errors produced during model execution are delivered to the central CGSI error services.
Refer to chapter NO TAG for more information about the CGSI error handling.

CSS Startup

Frame duration: 0.200 seconds”);

Copyright per DIN 34

CSS Kernel started
CSS Version: CGS/CSS Version x.x from <date>");

Buffer overflow: too many CSS events to be stored in the Test Result DataBase
(TRDB), 10 events dropped, 0 events stored
Detecting Unit: LOGGER_SPECIFIC_USERS_EVENT, Faulty Unit: CSS_EVT_HANDLER");

CMAS software successfully started
Detecting Unit: CTU, Faulty Unit: CMAS

Could not launch CSS event handler

CSS event handler started

CSS_EVT_HANDLER: Initialization completed

CSS_EVT_HANDLER: could not get associated sessionname, use default session instead
Status returned: SESSION_NOT_FOUND

CSS_EVT_HANDLER: shutting down...

CSS part of execution session
Name of session: <name>

CSS Kernel shutting down

CSS Logger_BE: Archive messages read: 2

CSS Logger_BE: Creating archive file
File: <unix_pathname>

CSS Logger_BE: Creating data set
File: <unix_pathname>

CSS Logger_BE: Creating data set statistics file",
File: <unix_pathname>

CSS Logger_BE: Creating logfile
File: <unix_pathname>

CSS Logger_BE: finished

CSS Logger_BE started

New Statevector loaded
Name of Statevector: <unix_pathname>

Copyright per DIN 34

D-3.4.2.1 Error messages produced by DB Browser, MOCS and ICP

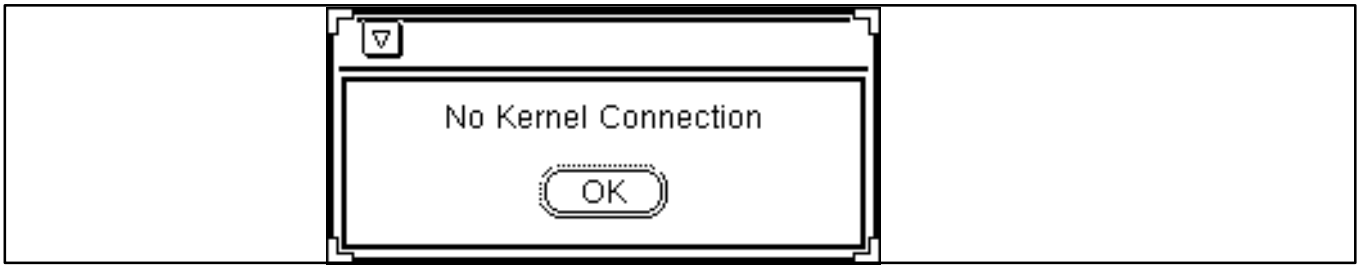


Figure 17 : *MOCS message window*

There is a special feature used by the MOCS/ICP error handling procedure. Each involved component adds one part of the error string, the result a an error string of unpredictable length. It is not possible to include all possible because the number of combinations is unlimited. An example of the resulting error string is given in Figure 18.

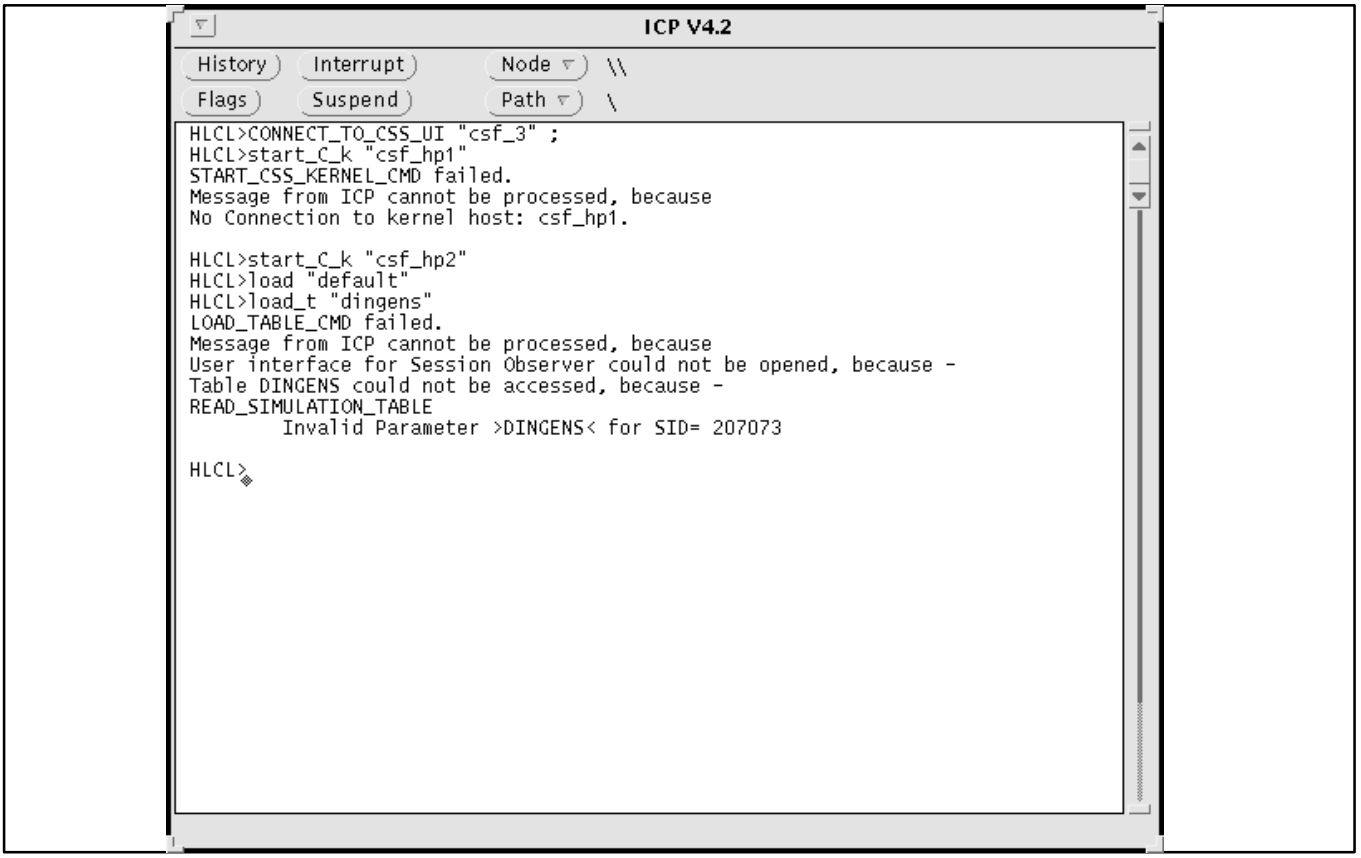


Figure 18 : *A concatenated error string is displayed in the ICP window*

Copyright per DIN 34

D-3.4.2.2 DB Browser Messages

C

Can not access model in database.

Contact your system administrator.

D

Database crashed.

Contact your system administrator.

DB server does not respond. Try again.

Make it so and contact your system administrator if it fails permanently.

E

Environment variable TWO_TASK is not set.

It is not possible to connect to the database. CSS was not installed properly.
Contact your system administrator.

F

First select a library in side the database browser.

User error. Select a library first.

I

In order to load the model under the selected user environment, all invalid function references have to be updated in the Database. Currently the model is locked against editing by another session.

The model is in use. Try again later.

In order to load the model under the selected user environment, all invalid function references have to be updated in the Database. You have not got the necessary edit permissions.

It is impossible to open the model with the currently selected CCU. Select a CCU which fits to the model references.

M

Model creation failed.

Try again, look for further error messages.

Copyright per DIN 34

Model removal failed.

Try again, look for further error messages.

N

<name> has been deleted or renamed.

A wrong model name is entered in a dialogue window. Type a correct i.e. existing models name.

O

Operation aborted: Model has been deleted or renamed.

There is no model with the given name. Check the spelling.

Operation aborted: Selected library is locked.

Another user is working with the selected library. Try again later.

Operation aborted: Selected model is locked.

Another user is working with the selected model. Try again later.

Operation aborted: There are still background jobs active.

Try again later.

P

Paste failed.

Try again, look for further error messages.

Paste failed: Copied model possibly has been deleted or renamed.

There is no model with the given name. Check the spelling.

Please close all open editors on the buffered model first.

Make it so.

Please close all open windows first.

Close the window(s), then repeat the command.

Copyright per DIN 34

Q

Quit aborted: Please close all open windows first.

An attempt is made to quit the DBB while there is at least one open window. Close the window(s).

Quit aborted: There are still background processes active.

Try again later.

R

Rename failed: Model possibly has been deleted or renamed.

There is no model with the given name. Check the spelling.

S

Simulation table removal failed.

Try again.

State vector removal failed.

Try again.

T

Table creation failed.

Try again, look for further error messages.

There is nothing to paste.

First copy the object, then paste.

There is no table to paste.

First copy the table, then paste.

U

Updated inconsistent model list.

The selected model does not exist any more, it is not possible to perform the desired operation.

The CCU you selected does not fit to the model. Check your selection.

Updating of invalid function references would exceed the maximum levels of decomposition.

The currently selected CCU doesn’t fit to the model references. Choose another CCU.

Copyright per DIN 34

D-3.4.2.3 MDE Error Messages – created during model editing

A

A record has to consist of at least <number> components.

Increase the record size accordingly.

AIL code contains the illegal keyword <name>.

Remove the illegal keyword.

AIL–reserved words must not be used as names.

Rename the item accordingly.

All engineering units must comply with AIL syntax.

Check the units (spelling).

All names must comply with AIL syntax.

Naming conventions are

* the first character must be a letter
* any subsequent characters must be letters, digits, or the underscore ('_')
* two underscores cannot occur together, nor can a model name end with an underscore
* a single space between letters and/or digits will be replaced by an underscore

All names must comply with AIL identifier syntax.

see above

All pathnames must comply with Database pathname syntax.

The Database pathname syntax is as follows:

* the first sign must be a backslash ('\')
* node names are separated by a backslash
* a node names consists of letters, digits or the underscore ('_')

All positions at the actual edited composite are occupied.

First enlarge the function block, then try again.

Copyright per DIN 34

All positions at the function block are occupied.

First enlarge the function block, then try again.

Attention: Name of pasted interface item has been changed.

This is not an error message. Note that you can't have two interface items with identical names assigned to the same object.

B

Both value receivers are already connected to different producers. Disconnect one and try again.

Make it so.

C

Can not access desktop directory <name>.

There is no directory with the given name. Check the spelling.

Can not access Interleaf configuration files.

Contact your system administration. Error caused by defective installation.

Can not access model in database.

There is no model with the given name. Check the spelling.

Can not apply <vector or matrix> of still undefined type.

First define the data type, then define the vector or matrix components.

Can not find a position for a new interface item.

Enlarge the function block.

Can't connect a MEASUREMENT to a MODEL INPUT.

You selected a wrong onboard item type – try stimulus.

Can't connect a STIMULUS to a MODEL OUTPUT.

You selected a wrong onboard item type – try measurement.

Can't connect incompatible interface items.

The type definitions of input and output do not fit together. Check the type definitions.

Copyright per DIN 34

Can’t edit interface item that is connected to an onboard item.

First disconnect the input/output, then edit it.

Compilation error.

Refer to section D-2.2.1.3 for detailed information.

Connect one of theses value receivers to a value producer first. Then try again.

Make it so.

Choose ”edit” to edit this row.

Make it so.

Choose ”toggle” to edit this row.

Make it so.

Column <number> has different action part.

It is not possible to minimize the matrix in the decision table because the column has a different action part. Select another column or remove the action part.

Copy/paste references from a non–frozen library is restricted to models in the same library.

That’s the way it is.

Connection to model interface: <name> is not onboard compatible.

Check the data type for inconsistencies.

CSS–reserved words must not be used as names.

Change the name.

Currently the model is locked for editing.

The model is in use, wait e few minutes then try again.

Currently the state codes are not defined.

First define the state codes, then try again.

Copyright per DIN 34

D

Decision table contains the illegal keyword <name>.

Remove the illegal keyword.

Desktop directory not specified.

Contact your system administrator. Installation fault.

Directory <name> does not exist.

Check your input (spelling).

Don’t connect value producers. (Please refer to the MDE–GL Manual.)

It is not possible to connect two outputs. In any case, refer to the CGS User Manual, (chapter: Model Development)

E

Error: Initial value is out of range.

Ignore, ranges have no meaning in the current implementation.

Error: <name> does not exist.

The model name does not exist. try again with a valid name.

Error: <name> exceeds the limit of <number> characters.

Rename the object.

Error: <name> identifies a directory.

You typed a directory name instead a model name. Correct the input.

Error: No read permission for <name>.

Copy the model, then read it or ask the model owner for giving you the read permission.

Error: Selected Interface item is connected.

First cut the connection, then delete the interface item.

Error: The state code size is restricted to <number> characters.

Shorten your state code.

Copyright per DIN 34

Error: Value is out of range.

Correct the value.

Errors in configuration.

Read the following error report for more information.

Execute create grouping link inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute create input inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute create output inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

Execute paste inside the graphic subview.

Move the mouse pointer into the graphic subview and repeat the command.

External referencing is restricted to functions in frozen libraries.

You can't make a reference to the desired library because it it not frozen.
Freeze it if possible or copy the desired function with the **copy–>normal** command.

F

Fatal Implementation Error !!!

This error should not occur.

File named <name> is an existing directory.

Wrong input. The name already exists.

First select the desired composite function block inside the hierarchy browser. Then try again.

Don't care about this message. It appears as soon as the hierarchy window is opened.

I

I/F items of record type may not become attached to the model interface.

Copyright per DIN 34

The record data type may be used only within the model.

Impossible to reduce size (marked object or one of its connections outside desired area.

Move your function blocks until they fit into the desired size.

It is not allowed to edit a referenced subtree.

The language elements you try to edit are copied by reference.
Use the **copy–>normal** command instead, then edit the items.

L

Local variables contain the illegal AIL keyword <name> .

Remove the AIL keyword from the local variables definition.

M

Macro definitions contain illegal AIL keyword <name> .

Remove the AIL keyword from the macro definition.

Model internal references are not allowed.

That's the way it is. Use the **copy–>normal** command.

Model is currently locked for editing.

It is not possible to perform the last initiated action. Try again later.

Model is locked.

It is not possible to perform the last initiated action. Try again later.

N

<name> already identifies another component.

Choose a different name.

<name> already identifies another interface item of parent function.

Choose a different name.

<name> contains unconnected reference points and can not be copied by reference.

Either connect the unconnected reference points and then try again or choose the **copy–>normal** command.

Copyright per DIN 34

<name> does not conform with AIL syntax.

Check your input. Note that AIL syntax is Ada syntax.

<name> is locked against inspecting.

If this error occurs report it via SPR (Software Problem Report) immediately to DASA/RI.

<name> is locked against editing.

The model is in use. Try later.

<name> selected interface items cannot be cut. Maybe they are connected over hierarchy level borders.

Check your implementation and cut low level connections first.

No connection line selected. Operation aborted.

You didn't move the mouse pointer to a connection line and click where to split the line. Try again.

No connection possible in this position. Move objects and try again.

Make it so.

No functions selected.

First select a function.

No write permission in directory <name>.

Try again with your own directory.

Not all names could be changed.

The **unify** command could not change all names. Check the implementation.

O

Only I/F items of onboard type may become attached to the model interface.

The selected input/output has a data type which can be used only within the model.

Only no matter fields may be expanded.

Select the 'no matter' field to expand the decision table. If there is no 'no matter' field the table has the maximum size.

Copyright per DIN 34

Operation aborted.

Try it again.

Operation aborted – would result in connection cycle.

It is not allowed to implement asynchronous chains, break the chain.

Operation aborted: Choose a new name to break the signal.

Your input is the name of the selected signal, type a new name to break the signal.

Operation aborted: Copied functions contain references into libraries not included in the model's user environment.

The selected CCU does not contain the referenced library or libraries. Go back and select an appropriate CCU.

Operation aborted: Initial value <number> would be out of range for engineering unit <name>.

Check initial value and engineering value – doesn't fit together.

Operation aborted: No CTG host selected.

Select a host in the pop–up menu.

Operation aborted: Pasting this interface item of RECORD type would lead to name conflicts.

Check the contents of the record for name conflicts.

Operation aborted: Record structure would lead to naming conflicts.

Check the record structure for duplicate names.

Operation aborted: The AIL code contains the following lines that exceed the limit of <number> characters.

Split the AIL statements into several lines.

Operation aborted: The combination of selected functions and selected filters will not produce any output.

Correct the filter setting and/or the function selection.

Operation aborted: The composite function contains unconnected reference points and can not copied by reference.

Copyright per DIN 34

Choose the **copy–>normal** command or correct the function block you want to reference.

Operation aborted: The output device is undefined.

Select either a TPS document or a printer as output device.

Operation aborted: No printer server selected.

Select the printer server (at least **default**). If there is no entry in the pop–up menu you found an installation error.

Operation aborted: would result in too many levels of decomposition.

The maximum levels of decomposition is 16.

P

Paste buffer contains function blocks.

It is not possible to paste a function block into an interface editor window.

Paste buffer is empty.

First copy something then paste.

Please close the MOCS window on <name> first.

Make it so.

Please close all open editors on <name> first.

Make it so.

Please select one or more Function Blocks for this operation and try again.

Make it so.

Position of last cuttet I/F item is already occupied.

Increase the size of the function block.

R

Referencing copied functions in this composite would result in loop.

Check your implementation for asynchronous loops.

Copyright per DIN 34

Rename the top level function inside the Database Browser.

Move the mouse pointer into the DBB window and repeat the command.

S

Select exactly one object for this operation.

Make it so.

Select exactly two objects for this operation.

Make it so.

Selected interface item is already connected to selected onboard item

No error – the desired connection is already established.

Selected item does not represent a Columbus onboard item.

Check the item definition, may be it is undefined.

Selected onboard item is incompatible to selected model interface item.

Check the item definition and the model input/output definition, maybe there is an undefined item.

Sorry, this function is not implemented yet.

What a pity !

Sorry, to a top level function are not supported..

What a shame !

Spaces have been replaced with '_'.

Ada syntax does not allow names with spaces inside.

T

The frame has to be an integer value between <min–number> and <max–number>.

Select the frame within the given limits.

The global symbol is connected to onboard. Cannot accept non onboard types.

Connect only to onboard types.

Copyright per DIN 34

The interface of <name> still contains grouping links. You have to remove all of them prior to converting a composite to an atomic function block.

Do not use the convert function. Create a new function block if you need one.

The last executed command cannot become undone.

Try it manually.

The limit of <number> record components is already reached.

Size the record within the limits.

The maximum number of actions is limited to <number>.

Reduce the actions in the decision table to the given number.

The maximum number of conditions is limited to <number>.

Reduce the conditions in the decision table to the given number.

The minimum matrix size is <number> rows and <number> columns.

Increase the number of elements.

The minimum number of record components is <number>.

Increase the number of elements.

The minimum vector size is <number>.

Increase the number of elements.

The model has not been modified since last saving.

The **save** operation will not be performed.

The model's user environment does not contain the selected library of onboard items.

Go back and select the appropriate CCU (with the correct onboard items).

The name was too long and has been truncated to the legal length of 16 characters.

If you don't like the truncated name, delete the object and create a new one.

Copyright per DIN 34

The number of elements is limited to <number>.

Reduce the number of elements.

The objects to paste don‘t fit with the size of the Composite Function Block. First choose a larger size then try again.

Make it so.

The referenced function is located in a library not visible in the current user environment.

The reference can’t be resolved. Select a CCU which contains the referenced function library.

The rotated object won’t fit with the size of the Composite Function Block. First choose a larger size then try again.

Make it so.

The selected definition point is unconnected.

It is not possible to show any references because the definition point is unconnected.

The selected interface item is connected to the interface of a reference function. You are not allowed to modify reference functions.

Replace the reference function by a copy of that function. Then try again.

The selected reference point is unconnected.

It is not possible to show any references because the reference point is unconnected.

The selected signal entry is unconnected.

It is not possible to disconnect this signal because it’s already unconnected.

The selected value receiver is connected to another producer. Disconnect it before making another connection.

It is not allowed to connect an input with two different outputs.

The type has to be defined in order to set the initial value.

First define the type, then set the initial value.

The type of a selected block could not be changed.

It is not possible to change the type. Delete the block and create a new one with the desired type.

Copyright per DIN 34

There are no objects to paste.

First copy or cut an item, then use the paste command.

There are no type inconsistencies in the selected global symbol.

Enjoy, everything is correct.

There is already an open AIL Editor.

You can’t get two editor window on the same object.

There is already an open Composite Editor.

You can’t get two editor window on the same object.

There is already an open Composite I/F Editor.

You can’t get two editor window on the same object.

There is already an open Documentation Tool.

You can’t get two documentation tools on the same object.

There is already an open Hierarchy Browser.

You can’t get two browser on the same object.

There is already an open Icon Editor.

Note that there is already an open editor window, close one.

There is no appropriate parent object for the new interface item.

This means ’All positions at the function block are occupied.’ Enlarge the function block and try again.

There is no column with an appropriate constellation of conditions.

It is not possible to minimize the matrix in the decision table. Select another condition field.

There is no icon to paste.

First **copy** the icon, then use the **paste** command.

Copyright per DIN 34

There is no onboard reference specified.

Select an onboard reference first.

There is no type compatible Definition Point.

You can only connect Reference and Definition points of the same type.

There is no unconnected Signal.

Disconnect a signal before you try to re–connect it.

These object are already connected.

Disconnect other connections, then try again.

This code generation process is still busy; the model is locked against saving.

Code generation can take several hours, watch the process and repeat your command as soon as the code generation is finished.

THIS ERROR SHOULD BE IMPOSSIBLE: <name>.

If this error occurs report it immediately (write a Software Problem Report).

This feature is currently not implemented.

What a pity !

This global symbol is connected to interface item(s) of reference function(s). The set of types had to be restricted to the type of the interface item(s) of the reference function(s).

Replace the reference function by a copy of the function. Then try again.

This global symbol is connected to multiple type incompatible interface items of reference functions. You are not allowed to modify reference functions.

Check if you use the referenced function block correctly.

This global symbol is referenced in a referenced subtree and can not be renamed.

You can delete the subtree and choose the **copy–>normal** function to copy the subtree. After that a rename is possible (but not recommended).

This interface item can only be attached to composite function blocks.

Make it so.

Copyright per DIN 34

This logical grouping does not contain any signals.

First define signals, then connect them.

This model is still busy and locked against editing.

Wait a few minutes, then repeat the last command.

This model is still busy and locked against printing.

Wait a few minutes, then repeat the last command.

This model is still busy and locked against saving.

Wait a few minutes, then repeat the last command.

This name already identifies another object. Choose a different name and try again.

Make it so.

This signal is referenced in a referenced subtree and can not be renamed.

You can delete the subtree and choose the **copy–>normal** function to copy the subtree. After that a rename is possible (but not recommended).

Y

You can’t add interface items to global symbols. Please refer to the MDE–GL Manual.

Reference points and definitions points have exactly one input/output. You can’t add another input/output.
In any case, refer to the CGS User Manual (chapter: Model Development).

You can’t add interface items to parameter blocks. Please refer to the MDE–GL Manual.

Parameter blocks exactly one output. You can’t add output.
In any case, refer to the CGS User Manual (chapter: Model Development).

You have to select an onboard item inside the Database Browser.

Press the DMS button in the DBB to switch to the onboard item selection.

Copyright per DIN 34

D-3.4.2.4 Compilation errors from the CSS runtime system (CTG)

CTG is still in the development stage so changes are expected.

A

ADALIB_CREATION_ERROR: cannot create Ada library <name>; see error file <name>.

Please read the error file which is located in your $HOME/????. In this file you find a detailed error description.

C

CTG_CALL_ERROR: Wrong number of arguments; Usage: css_ctg_server <port_file_name>.

Please write a Software Problem Report (SPR) immediately.

D

DIRECTORY_CREATION_ERROR: cannot create directory <name>.

Please write a Software Problem Report (SPR) immediately.

E

ENVIRONMENT_ERROR: The following environment variable is not set: <name>.

Please contact your System Administrator. CSS was not installed correctly.

F

FILE_ACCESS_ERROR: <error_code> raised during accessing the file <name>.

Please check your installation for quota problems or missing mounts to other machine/programs.

FILE_ALREADY_EXISTS_ERROR: The file or directory <name> already exists.

Please write a Software Problem Report (SPR) immediately.

FILE_COPY_ERROR: cannot copy file or directory <name> to file resp directory <name>.

Please check your installation for quota problems.

FILE_CREATION_ERROR: cannot create file or directory <name>.

Please write a Software Problem Report (SPR) immediately.

FILE_DELETION_ERROR: cannot delete file or directory <name>.

Please write a Software Problem Report (SPR) immediately.

Copyright per DIN 34

FILE_NOT_FOUND_ERROR: The file <name> does not exist.

Please write a Software Problem Report (SPR) immediately.

FILE_RENAMING_ERROR: cannot move file or directory <name> to file resp directory <name>.

Please write a Software Problem Report (SPR) immediately.

I

INCONSISTENT_SYSTEM_ERROR: the client and the server are incompatible: version_nr. of client: <number>; version_nr of server: <number>.

Please contact your System Administrator. CSS was not installed correctly.
Please write a Software Problem Report (SPR) immediately.

INTERNAL_ERROR: <error_code> raised in subprogram <name>.

Please write a Software Problem Report (SPR) immediately.

INVALID_SERVICE_ERROR: provided and required requests are incompatible: provide request <name>; required requests <name>.

Please write a Software Problem Report (SPR) immediately.

N

NO_CONNECTION: cannot establish connection between server and client; or connection to client broken.

This is a configuration problem. Please check your installation for network problems or missing mounts to other machines.

P

PORT_CREATION_ERROR: cannot create new port.

This is a configuration problem. CSS requires port numbers in the range from 8000 to 8999.

Copyright per DIN 34

D-3.4.2.5 MOCS error messages

- *Note that the % sign will be replaced by the actually used parameter.*
- *Note that the – sign denotes that the error string will be concatenated with another error string.*

A

Are you sure, that you want the scale steps to be greater than the range ?

The scale steps will not be visible. Check your input.

C

Cannot associate definition to definition table: %.

The user tried to log an item or trace a function block. He/she has to specify a Simulation table to associate the definition with.
Correct the error by either loading the specified table table first or specify a different table.

Cannot close CSS User Interface, because – %.

Check the complete error message for the possible reason.
Try the HLCL STOP_CSS_UI command.

Cannot connect to kernel, because – %.

Check the complete error message for the possible reason.

Cannot interpret input stream: % was not expected in state %

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot load Kernel/Mapping Table from DB.

Check the database connection. If there is no problem , this is an internal error message.

Cannot process event % because – % is not defined.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot read complete records.

The user referenced a CSS record type variable in HLCL. Only record elements can be referenced. Change your HLCL command accordingly.

Copyright per DIN 34

Cannot read mapping table because –

Check the complete error message for the possible reason.
Please contact your System Administrator. It might possibly be an incorrect CSS installation.

Cannot scan % in specification, because –

Internal error message. Please write a Software Problem Report (SPR) immediately.

Cannot start kernel, because %.

Check the complete error message for the possible reason.

Command could not be executed, because – %

Check the complete error message for the possible reason.

Command Interpreter could not be started, because – %.

Check the complete error message for the possible reason.
The reason could be a communication error raised during the connection building phase, e.g. the port number is undefined or still in use.

Command was successfully submitted to simulator. You can view it by pressing SHOW COMMANDS.

This is not an error message.

Connection timeout expired for connection to %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Connection to % broken.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Could not determine inconsistent entries, because –

Check the complete error message for the possible reason.

Could not encode % in a string, because –

Check the complete error message for the possible reason.

Copyright per DIN 34

could not write % to database, because –

Check the complete error message for the possible reason.

could not write % to file, because –

Check the complete error message for the possible reason.

CSS cannot connect to Kernel because –

Check the complete error message for the possible reason.

D

Daemon for ICP Connection could not be started, because –

Check the complete error message for the possible reason.
Perhaps because the port name used was undefined or the port number was still occupied bey another process.
If the port number is still in use (see rest of the error message), retry later.

Database Identifier (SID) could not be determined.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Definitions could not be stored, because %

Check the complete error message for the possible reason.

E

Error accessing kernel identification, because – %

Check the complete error message for the possible reason.

F

Function cannot be traced because %

Check the complete error message for the possible reason.

H

HLCL command is not allowed in this system state.

Check in the User Manual, in which context this command is allowed.

Copyright per DIN 34

I

Identifier Mapping Table File % not found.

Check the database connection. If the connection is ok, this is an internal error message.

Illegal step count % entered. Step count must be a numeric value.

The input must be a number greater 0.

Illegal step count % entered. Step count must be positive.

The input must be a number greater 0.

Illegal % syntax.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Incompatible SW versions of MOCS and %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Invalid input, try again.

Check your input for correctness.

Item cannot be locked, because %

Check the complete error string for the specific reason.

Item named % cannot be monitored or logged.

Ensure that the item is either external (top level I/O) , atomic or an output item.

Item named % may not be assigned values.

This value does not allow assignments. Check the data type and ensure that the item is either a top level input or a model output.

K

Kernel crashed during start up, because – %

Copyright per DIN 34

Check the complete error message for the possible reason.

Kernel did not respond to command in time.

Try again.

Kernel did not start in time.

Try again.

Kernel Interface File could not be accessed, because – %

Check the complete error message for the possible reason.

Kernel is disconnected.

A command was given which is not allowed in the current model state.

Kernel Startup Warning: %

Check the complete error message for the possible reason.

L

Local command action could not be performed, because – %
!!! attention: system may be inconsistent, stop and restart !!!

Check the following complete error message for a description of inconsistencies. Then stop and restart if necessary.

Logfile and Archivfile can be found at the locations: % and %

This is not an error message. It simply informs the user where to find the log– and archive–files.

M

Message coming from % with the operation code % cannot be interpreted.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Message could not be encoded, because –

Check the complete error message for the possible reason.

Copyright per DIN 34

Message could not be send to kernel, because – %

Check the complete error message for the possible reason.

Message from ICP cannot be processed, because %

Check the complete error message for the possible reason.

Minimum value may not be equal to maximum value.

Check your input.

Minimum value must be less than maximum value.

Check your input.

More than one Kernel for the specified model is running.

Note that the connection to a Kernel can only be established if exactly one Kernel is running on the host. Make sure that only one Kernel is running.

N

Negative virtual factor % is illegal input.

Check your input for correctness. Then try again.

No CMAS host specified.

If you want to start CMAS, you have to specify a host machine for CMAS.

No connection to kernel host: %

Check the complete error message for the possible reason.
One possible reason: a host name has been specified, but before starting the Kernel on that host, the existence of the host in the network is tested and the test failed.

No item named % could be found.

The model does not contain any item with the specified name. Check your parameter list for correctness.

No item with destination % selected.

The user tried to remove a tracing definition from an item which is not traced.

Copyright per DIN 34

No kernel for the specified model is running on host %.

Check your command for correctness. If necessary start the simulator on the desired host and try again.

No kernel host specified.

The user pressed the kernel start button, but did not specify the host on which the kernel should be started. Try again.

No kernel is configured for %

It is not possible to start the desired model. Check whether the model is configured for the corresponding machine type. Re–configure the model if necessary.

No open connection message from % received. Received OP–code % instead.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No path for RID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No path for SID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No privilege to log.

The user has no log privilege. Use the HLCL command **GRANT** to grant he specified privilege or choose the screen as destination for snapshots/traces.

No RID found for SID %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No RID for path % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No rule is specified for state %.

Copyright per DIN 34

Internal error message. Please write a Software Problem Report (SPR) immediately.

No SID for RID % found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

No simulation table selected.

First select a simulation table, then repeat the command.

Non–numeric virtual factor % entered.

Check your input for correctness.

P

Pathname does not contain the correct model library.

Check the pathname for correct input.

Pathname % does not exist.

Check the pathname for correct input.

Previous command is still in execution.

Wait until the command is finished, then try again.

Programming error in ItemIdentificator: Illegal item reference.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Programming error in ItemIdentificator: No item reference available.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Protocol error in message with operation code %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Pulse type variables can only be set.

It not possible to reset a pulse variable.

Copyright per DIN 34

R

Received unexpected reply with identification: %

Internal error message. Please write a Software Problem Report (SPR) immediately.

S

Scanning error occurred at position %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Semantics error, unsuccessfully sent message %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

SID for pathname % not found.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Simulation command % is not known to the message encoder.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Snapshot of value could not be taken, because %

Check the complete error message. Probably the interface item identification was not legal.

Specified item % is already locked.

The user tried to log an item, which is already locked. Check your input for the correct item name.

Specified item % is not yet already logged.

The user tried to remove a logging definition which does not exist.

Specified item % is not yet already monitored.

The user tried to remove a monitoring definition which does not exist.

T

Table % could not be accessed, because – %

Copyright per DIN 34

Check the complete error message for the possible reason.

The HLCL Command Interpreter send ERROR: %

This is not a MOCS error.
Report the error message to the HLCL maintenance team.

The HLCL Command Interpreter send FATAL: %

This is not a MOCS error.
Report the error message to the HLCL maintenance team.

The item % is already monitored with different monitoring parameters.

The user tried to monitor an item, which is already monitored elsewhere on the screen. Check your input for the correct item name.

The function block % is already traced to the %.

The user tried to trace a function block, which is already to the specified destination. To change the destination first remove the trace option, then try again.

The service point directory % cannot be accessed, because %

Check the complete error message for the possible reason.

The service point file % cannot be interpreted, because %

Check the complete error message for the possible reason.

The specified function block % was not traced to the %.

The user tried to remove a tracing definition, which was not traced to the specified destination. Try again with the correct destination.

The value returned on the read item command was not the value of the expected model item.

Internal error message. Please write a Software Problem Report (SPR) immediately.

This branch may not be passed: %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Copyright per DIN 34

This command may not be cancelled.

Only time tagged commands can be cancelled. The wait command may not be cancelled.

This path may not be passed: %

Internal error message. Please write a Software Problem Report (SPR) immediately.

Tried to store a value in a ConditionEvaluation, which is not a value holder.

Internal error message. Please write a Software Problem Report (SPR) immediately.

U

Unexpected event with operation code % received from %.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Unknown character % (%) was found while scanning.

Internal error message. Please write a Software Problem Report (SPR) immediately.

Unknown execution status % of acknowledgement.

Internal error message. Please write a Software Problem Report (SPR) immediately.

User interface for % could not be opened, because – %.

Check the complete error message for the possible reason.

V

Value type is incorrect.

Check your input for correctness.

Variables of type Pulse/Burstpulse cannot be monitored or logged.

If the user wants to ensure, that the event occurred, the event must be registered in a counter for instance. A counter must be included explicitly in the model implementation for this purpose.

Variables of type Pulse cannot be logged. Pulse type variables are excluded from the command.

Copyright per DIN 34

Check your input.

Variables of type Pulse cannot be monitored. Pulse type variables are excluded from the command.

Check your input.

%

% is not an atomic function block.

The operation requires an atomic function block as parameter. Check your parameter list.

% is not an interface item.

The operation requires a top level input or a model output as parameter. Check your parameter list.

% must be specified.

Check the first part of the error message for the item identification.

...

... see MOCS console for kernel error messages.

This error message occurs whenever the execution of a command by the simulator was not successfully terminated. Check the MOCS console window for a detailed error message.

Copyright per DIN 34

D-3.4.2.6 Runtime Error Messages created during model execution

A

ADA I/O Error: STATUS_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to operate upon a file that is not open, and by an attempt to open a file that is already open.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: MODE_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to read from, or test for the end of, a file whose current mode is OUT_FILE, and also by an attempt to write to a file whose current mode is IN_FILE.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: NAME_ERROR.

This exception is raised by input–output operations.
The exception is raised by a call of CREATE or OPEN if the string given for the parameter NAME does not allow the identification of an external file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: USE_ERROR.

This exception is raised by input–output operations.
The exception is raised if an operation is attempted that is not possible for reasons that depend on the characteristics of the external file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: DEVICE_ERROR.

This exception is raised by input–output operations.
The exception is raised if an input–output operation cannot be completed because of the malfunction of the underlying system.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.
Report this error to your responsible system administrator immediately.

ADA I/O Error: END_ERROR.

This exception is raised by input–output operations.
The exception is raised by an attempt to skip (read past) the end of a file.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

Copyright per DIN 34

ADA I/O Error: DATA_ERROR.

This exception is raised by input–output operations.
The exception may be raised by the procedure READ if the element cannot be interpreted as a value of the required type. This exception is also raised by a procedure GET if the input character sequence fails to satisfy the required syntax, or if the value input does not belong to the range of the required type or subtype.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA I/O Error: LAYOUT_ERROR.

This exception is raised by input–output operations.
The exception is raised in text input–output if a value returned exceeds COUNT'LAST.
The exception is also raised on output by an attempt to set column or line numbers in excess of specified maximum line or page length, respectively.
It is also raised by an attempt to PUT too many characters to a string.
Refer to the Programming Language Ada Reference Manual section 14.4 for more Information.

ADA Runtime Error: CONSTRAINT_ERROR.

This exception is predefined in the Ada language.
The exception is raised in any of the following situations:
upon an attempt to violate a range constraint, an index constraint or a discriminant constraint;
upon an attempt to use a record component that does not exist for the current discriminant values;
and upon an attempt to use a selected component, an indexed component, a slice, or an attribute, of an object designated by an access value, if the object does not exits because the access value is null.
Refer to the Programming Language Ada Reference Manual for more Information, then
check your model for possible rule violations.

ADA Runtime Error: NUMERIC_ERROR.

This exception is predefined in the Ada language.
The exception is raised by the execution of a predefined numeric operation that cannot deliver a correct result (within the declared accuracy for real typed);
this includes the case where an implementation uses a predefined numeric operation for the execution, evaluation, or elaboration of some construct.
Refer to the Programming Language Ada Reference Manual for more Information, then
check your model for possible rule violations.

ADA Runtime Error: PROGRAM_ERROR.

This exception is predefined in the Ada language.
The exception is raised upon an attempt to call a subprogram, to activate a task, or to elaborate a generic instantiation, if the body of the corresponding unit has not yet been elaborated.
This exception is also raised if the end of a function is reached;
or during the execution of a selective wait that has no else part, if this execution determines that all alternatives are closed.
Finally, this exception may be raised upon an attempt to execute an action that is erroneous, and for incorrect

Copyright per DIN 34

order dependences.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: STORAGE_ERROR.

This exception is predefined in the Ada language.
The exception is raised in any of the following situations:
when the dynamic storage allocated to a task is exceeded;
during the evaluation of an allocator, if the space available for the collection of allocated objects is exhausted; or during the elaboration of a declarative item, or during the execution of a subprogram call, if storage is not sufficient.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: TASKING_ERROR.

This exception is predefined in the Ada language.
This exception is raised when exceptions arise during intertask communication.
Refer to the Programming Language Ada Reference Manual for more Information, then check your model for possible rule violations.

ADA Runtime Error: OTHER_ERROR.

Any other error which doesn't fit into the categories the the predefined runtime errors will be reported here.

ADA: Case selector undefined.

This is an internal ICD problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

C

Constraint Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU Exception raised during rendezvous in Timeout–Entry.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Copyright per DIN 34

CTU Exception raised during reception of ZERO_LOAD_INDICATOR–Response.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU Fatal–ERROR: Exception raised during initialization phase.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

CTU raised an exception while in 'selective accept'.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

E

Error allocating new memory.

This is a memory overflow problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Error during initialization: Shared–memory–Error.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Error during initialization: Missing acknowledge from Adaptation–System.

Check your configuration i.e. for missing executables.

F

Failure during CTU's initialization phase.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

N

Numeric Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

Copyright per DIN 34

P

Program Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

S

Storage Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

T

Tasking Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

O

Other Exception found in CTU.

This is an internal coding problem.
If this error occurs report it via SPR (Software Problem Report) to DASA/RI immediately.

R

Resuming monitoring.

This is not an error message – the monitoring was suspended because there are too many monitoring values. Now it's starting again.

T

Too many monitoring values – suspend monitoring.

This is an internal traffic problem. Too many monitoring items are send which cannot be handled by the receiver.
Change the simulation table by reducing the number of monitoring items and try again.

U

Unable to close logfile.

You have no permission to write a log file. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

Copyright per DIN 34

Unable to open archive file.

You have no permission to open an archive file. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

Unable to open logfile.

You have no permission to open an logfile. Check the contents of the environment variable CSS_LOG_DIR and change it accordingly or
Check the available free storage space on your account.

Copyright per DIN 34

D-4 Test Evaluation

D-4.1 TEV Error Report Basics

The minimum information given to the user is a context–driven message displayed in the frame footer of the concerned window after each user transaction. If an error occurs then error status will be added.

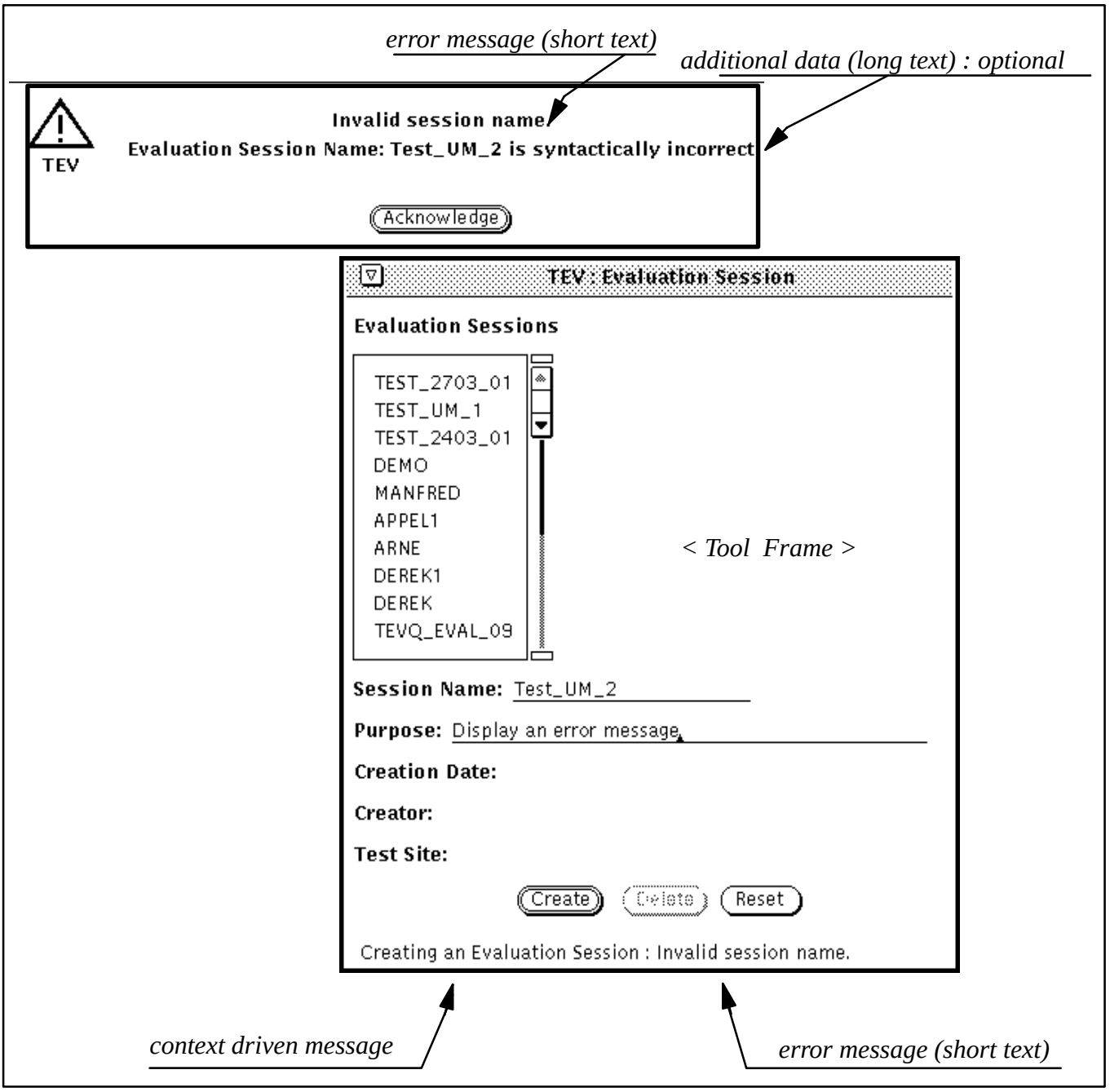


Figure 19 : *Error Report example in TEV*

D-4.1.1 DBS error messages displayed in TEV

Note that different DBS errors may be displayed with a unique TEV error message text.

Copyright per DIN 34

TEV reports the following errors for each of the DBS errror status :

DBS_ERROR	TEV_ERROR
INDEX_ERROR	TEV_INTERNAL_ERROR
LIST_NOT_NULL	TEV_INTERNAL_ERROR
SESSION_NOT_FOUND	SESSION_NOT_FOUND
INVALID_SESSION_NAME	INVALID_SESSION_NAME
EXISTING_SESSION	SESSION_ALREADY_EXISTING
SESSION_IS_USED	SESSION_IS_USED

A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not exclusively allocated to him.

SESSION_IS_ALLOCATED	SESSION_IS_ALLOCATED
SESSION_IS_ARCHIVED	SESSION_IS_ARCHIVED
SESSION_NOT_ARCHIVED	SESSION_NOT_ARCHIVED
SESSION_IS_EMPTY	SESSION_IS_EMPTY
SESSION_IS_ONLINE	SESSION_IS_ONLINE
RETRIEVING_NOT_AUTHORISED	SESSION_IS_ONLINE
DATA_ALREADY_ONLINE	DATA_ALREADY_ONLINE
DATA_PARTIALLY_ONLINE	DATA_PARTIALLY_ONLINE
DATA_NOT_ONLINE	DATA_NOT_ONLINE
EXISTING_FILE	FILE_ALREADY_EXISTING
FILE_NOT_FOUND	FILE_NOT_EXISTING
DATA_NOT_FOUND	DATA_NOT_FOUND
EXISTING_USER	EXISTING_USER
PRINTING_REROUTED	PRINTING_REROUTED

Copyright per DIN 34

DBS_ERROR	TEV_ERROR
FILE_IS_EMPTY	FILE_IS_EMPTY
CANCEL_NOT_AUTHORISED	CANCEL_NOT_AUTHORISED
JOB_CANCELLED	JOB_CANCELLED
FA_MEDIUM_NOT_AVAILABLE	FA_ERROR
FA_MMI_REQUEST_CANCELLED	FA_ERROR
FA_RESERVED_FOR_AUTO_ARCH	FA_ERROR
FA_DEVICE_NOT_ALLOCATED	FA_ERROR
ACTIVE_FA_JOB LIST_NULL	FA_ERROR
INSUFFICIENT_PRIVILEGE	INSUFFICIENT_PRIVILEGE
CONNECT_PROBLEM	SYS_ERROR
INIT_PROBLEM	SYS_ERROR
DBS_NOT_CONNECTED	SYS_ERROR
DBS_STOPPED	SYS_ERROR
DISCONNECT_PROBLEM	SYS_ERROR
SESSION_NOT_OPEN	SYS_ERROR
INACCESSIBLE_FILES	SYS_ERROR
FILE_SYSTEM_FULL	SYS_ERROR
POOL_SATURATED	SYS_ERROR
TABLE_SPACE_FULL	SYS_ERROR
UNKNOWN_PRINTER	SYS_ERROR
FILE_NOT_PRINTABLE	SYS_ERROR
SPOOLER_DISACTIVATED	SYS_ERROR
PRINTER_NOT_DEFINED	SYS_ERROR
NO_PRINTER_AVAILABLE	SYS_ERROR
UNKNOWN_ERROR	SYS_ERROR
NOT_ENOUGH_SPACE	SYS_ERROR

Copyright per DIN 34

NOT_CONNECTED_TO_EVAL	SYS_ERROR
FA_INTERNAL_PROBLEM	SYS_ERROR
FA_UNIX_PROBLEM	SYS_ERROR
FA_MMI_TIME_OUT	SYS_ERROR
FA_LIST_NOT_FOUND	SYS_ERROR
LIST_OVERFLOW	SYS_ERROR
COMMS_TIME_OUT	SYS_ERROR
UNKNOWN_PRIVILEGE	SYS_ERROR

Copyright per DIN 34

DBS_ERROR	TEV_ERROR
DBS_ORACLE_PROBLEM	SYS_ERROR
UNIX_PROBLEM	SYS_ERROR
DBS_INTERNAL_PROBLEM	SYS_ERROR
DBS_COMMUNICATION_PROBLEM	SYS_ERROR
BAD_SELECTION_CRITERIA	SYS_ERROR
EXISTING_FA_JOB	SYS_ERROR
FA_DRIVE_NOT_AVAILABLE	SYS_ERROR

Copyright per DIN 34

D-4.1.2 TEV Footer Messages

As described earlier in section D-4.1 the footer message in combination with the status may report an error. The footer message denotes the action to be performed, the status message says wether the action was successful or not and specifies the nature of the error. Corrective actions depend on the displayed combination of message and status.

The following messages may be reported in TEV footers :

Start TEV

Quit TEV

Start to initialise session

Start Events Logging Tool

Start Raw Data Dump Tool

Start Data Set Generation Tool

Start Statistics Generation Tool

Start Data Listing Generation Tool

Start Graph Tool

Loading a Definition

Saving the Current Definition

Updating the Definition

Printing the Current Definition

Resetting the Current Definition

Displaying the Current Definition

Executing the Current Definition

Printing a Definition

Getting the Definition List

Deleting a Definition

Clearing Definition

Overwriting a Definition

Loading the Current Result

Copyright per DIN 34

Saving the Current Result

Printing the Current Result

Displaying the Current Result

Printing a Result

Getting the Result List

Storing a Result

Renaming a Result

Copying a Result

Overwriting a Result

Deleting a Result

Getting the Parameter List

Selecting Parameters

Checking the Parameter Set

Inserting a Parameter

Formatting a Packet

Setting the output parameters

Fetching the Current Packet

Jumping to a Packet

Saving the Current Packet

Overwriting a Packet

Printing the Current Packet

Displaying the Current Packet

Selecting an Evaluation Session

Creating an Evaluation Session

Deleting an Evaluation Session

Confirm deletion of an Evaluation Session

Getting Evaluation Session List

Copyright per DIN 34

Getting Evaluation Session Info

Selecting an Execution Session

Creating an Execution Session

Deleting an Execution Session

Initialising Execution Sessions

Confirm deletion of an Execution Session

Getting Execution Session List

Getting Execution Session Info

Getting Data Set Info

Merging Data Set

Start Merge

Getting User Event List

Selecting User Events

List CCUS From MDB

Getting the online data

Getting archived sessions

Getting on line sessions

Archiving a session

Retrieving a session

Deleting on line data

Getting session data location

Deleting session data

Updating final archive window

Converting a Result

Importing a session

Exporting a session

Updating Import/Export window

Copyright per DIN 34

Getting job list

Removing job

Selecting default printer

Displaying job queue

Displaying SAS tool

Selecting a SAS

SAS Selected

Starting a SAS

Copyright per DIN 34

D-4.1.3 TEV Status Messages

As described earlier in section D-4.1 the footer message in combination with the status may report an error. The footer message denotes the action to be performed, the status message says wether the action was successful or not and specifies the nature of the error. Corrective actions depend on the displayed combination of message and status.

The following TEV status are reported with the message described.

TEV_STATUS	MESSAGE
OK	”Ok.”
RUNNING	”...”
TEV_INTERNAL_ERROR	”TEV Internal error.”
DBS_ERROR	”DBS error.”
MDB_ERROR	”MDB error.”
SYS_ERROR	”System error”
PRINT_ERROR	”Print error.”
ACCESS_VIOLATION	”Access violation.”
TOOLS_OVERFLOW	”Tools overflow.”
TOOLS_RUNNING	”Tools are running.”
LIST_OVERFLOW	”List Overflow.”
EXISTING_USER	”Existing User.”
TRUNCATED_LIST	”Truncated List.”
FILE_NOT_EXISTING	”File not existing.”
FILE_IS_EMPTY	”File is empty.”
FILE_NOT_SELECTED	”No file selected.”
NO_DIRECTORY_SELECTED	”No directory selected.”
DIRECTORY_NOT_EXISTING	”Directory not existing.”
DIRECTORY_ALREADY_EXISTING	”Directory already existing.”
NO_TARGET_FILE_SELECTED	”Target file not selected.”
NO_SOURCE_FILE_SELECTED	”Source file not selected.”

Copyright per DIN 34

SOURCE_FILE_UNDEFINED	”Undefined source file.”
DESTINATION_UNDEFINED	”Undefined destination.”
FILE_ALREADY_EXISTING	”File already existing.”
TOO_LONG_FILENAME	”Filename is too long.”
TOO_LONG_DIRNAME	”Directory name is too long.”
FILE_NOT_PRINTABLE	”File not printable.”
FILE_SYSTEM_FULL	”File system full.”
FILE_ACCESS_ERROR	”File access error.”
NOT_AN_ARCHIVE_FILE	”Invalid archive file.”
INVALID_SOURCE	”Invalid source.”
TOO_MANY_SEL_EXECUTION_SESSIONS	”Too many selected execution sessions.”
INVALID_SESSION_NAME	”Invalid session name.”
NO_EXECUTION_SESSION_SELECTED	”No execution session selected.”
NO_EVALUATION_SESSION_SELECTED	”No evaluation session selected.”
SESSION_NOT_FOUND	”Session not found.”
SESSION_IS_OPEN	”Session is open”
SESSION_IS_ALLOCATED	”Session is allocated”
SESSION_IS_USED	”Session is used”

A SESSION_IS_USED error will be reported if a user tries to delete, archive or export a session which is not exclusively allocated to him.

SESSION_IS_EMPTY	”Session is empty”
SESSION_IS_ARCHIVED	”Session is archived”
SESSION_NOT_ARCHIVED	”Session is not archived”
SESSION_IS_ONLINE	”Session is online”
SESSION_ALREADY_EXISTING	”Session already existing.”
JOB_CANCELLED	”Job cancelled.”
DATA_ALREADY_ONLINE	”Data already online.”
DATA_PARTIALLY_ONLINE	”Data partially online.”

Copyright per DIN 34

DATA_NOT_ONLINE	”Data not online.”
PRINTING_REROUTED	”Printing re–routed.”
CANCEL_NOT_AUTHORISED	”Cancel not authorised.”
INSUFFICIENT_PRIVILEGE	”Insufficient privilege.”
EMPTY_LIST	”List is empty.”
DATA_NOT_FOUND	”Data not found.”
NULL_SESSION_NAME	”Session name is null.”
WRONG_SELECTION	”Wrong selection.”
INVALID_CCU	”Invalid CCU.”
NO_INITIAL_TIME_FRAME	”Initial time frame not defined.”
NO_GLOBAL_TIME_FRAME	”Global time frame not defined.”
NULL_TIME_FRAME	”Time frame is null.”
TIME_FRAME_ERROR	”Error in time frame.”
TIME_FRAME_NOT_IN_INITIAL	”Time frame not included in initial time frame.”
TIME_FRAME_NOT_IN_OVERALL	”Time frame not included in overall time frame.”
INVALID_DATE_TIME	”Invalid date and time.”
LEXICAL_ERROR_IN_TIME_FRAME	”Lexical error in time frame.”
PATHNAME_ERROR	”Pathname error.”
INVALID_PARAMETER	”Parameter is invalid.”
DUPLICATE_PARAMETER	”Parameter is duplicated.”
INVALID_SOFTWARE_PARAMETER	”Software parameter is invalid.”
NO_PARAMETER_SELECTED	”No parameter selected.”
EMPTY_PARAMETER_SET_REMOVED	”An empty parameter set has been removed.”
NO_SOURCE_SELECTED	”Source is not selected.”
DATA_SET_GENERATION_ERROR	”Error during generation of data set.”
CALIBRATION_ERROR	”Calibration error.”
ARCHIVE_FILE_ERROR	”Archive file error.”

Copyright per DIN 34

SAMPLING_ERROR	”Sampling error.”
SID_ERROR	”SID error.”
TOO_MANY_PARAMETERS	”Too many parameters.”
NO_MORE_PACKET	”No more packet.”
ADU_NOT_EXISTING	”ADU not found.”
GDU_NOT_EXISTING	”GDU not found.”
DATA_ERROR	”Data Error.”
NO_CURRENT_ADU	”No current ADU.”
NO_CURRENT_GDU	”No current GDU.”
INCOMPLETE_PACKET_TYPE	”Incomplete packet type.”
INVALID_OUTPUT_FORMAT	”Invalid output format.”
OUT_OF_RANGE_PACKET_SIZE	”Packet size is out of range.”
WRONG_BYTES_PER_LINE	”Wrong number of bytes per line.”
EXISTING_DEFINITION	”Existing definition.”
NO_DEFINITION_EXISTING	”No existing definition.”
NO_DEFINITION_SELECTED	”No definition selected.”
NOT_EXISTING_DEFINITION	”Not existing definition.”
NO_CHANGE	”No change.”
NULL_DEFINITION	”Null definition.”
NULL_DEFINITION_NAME	”Null definition name.”
ERROR_ACCESSING_DEFINITION	”Error accessing definition.”
EXISTING_RESULT	”Existing result.”
RESULT_GENERATION_ERROR	”Result generation error.”
RESULT_TYPE_UNDEFINED	”Result type undefined.”
NO_RESULT_EXISTING	”No result existing.”
NO_RESULT_SELECTED	”No result selected.”
NOT_COMPLETE_RESULT	”Not complete Result.”

Copyright per DIN 34

NOT_EXISTING_RESULT	”Not existing result.”
NULL_RESULT	”Null result.”
NULL_RESULT_NAME	”Null result name.”
NOT_COMPLIANT_PARAMETER_LIST	”Not compliant parameter list.”
TRUNCATED_PARAMETER_LIST	”Truncated parameter list.”
MISMATCHED_PARAMETER_TYPES	”Mismatched parameter types.”
NOT_EXISTING_DATA_SET	”Not existing data set.”
NO_DATA_SET_SELECTED	”Not data set selected”
INSUFFICIENT_PARAMETERS_DEFINED	”Insufficient parameters have been defined”
LEXICAL_ERROR_IN_SCALING_STR	”Lexical error in graph scaling string”
INVALID_SCALING	”Error in graph scaling”
INVALID_NAME	”Error in graph name”
UNABLE_TO_CONNECT_TO_NEWS	”Unable to connect to NeWS server”
TOO_MANY_SESSIONS	”Merged data set would contain too many sessions”
SOURCE_MISMATCH	”Sets to merge contain parameter from different sources”
ENGINEERING_UNIT_MISMATCH	”Sets to merge contain parameter with different units”
VALUE_MISMATCH	”Sets to merge have value discrepancies”
INVALID_DATA_SET	”Invalid data set.”
INSUFFICIENT_USER_EVENTS_SELECTED	”Insufficient user events selected.”
OVERLAPPING_TIME_FRAME	”Overlapping time frame.”
DIFFERENT_CCUS	”Selected sessions have different CCUs.”
MERGE_SELECT_ERROR	”Data sets to merge must be selected on same time type”
MERGE_ORDER_ERROR	”Data sets to merge must be ordered by same time type”
NO_JOB_SELECTED	”Job not selected”

Copyright per DIN 34

SAS_NOT_EXISTING	”SAS not existing”
SAS_NOT_SELECTED	”SAS not selected”
SAS_NOT_EXECUTABLE	”SAS not executable”

D-4.1.4 TEV Error Messages

TEV error messages may come from different sources. In the following sections TEV error messages are listed with the corresponding error text of the source which reported the error.

**** MAIN PROGRAM ABANDONED — EXCEPTION ”TEV_PROCESS_LOCKED” RAISED**
—> Verify no other TEV is running on this node. Then remove lock file: rm /tmp/TEV.lock

ERROR (TDCS_RPI) INIT_TDCS_RPI: Exception raised
FATAL_ERROR (TDCS_RPI) Can’t initialize TDCS RPI
—> There is a problem with the TDCS (SVF specific) software
—> can be ignored on environments where SVF is not installed

ERROR : TEV Internal error.Current exception: X_ERROR_ACCESSING_FILE
Consequence : Current operation fails with unexpected error.
Recovery action : No specific recovery. See system administrator.
Debug information : EXECUTE_CURRENT_DEFINITION”);

ERROR : System error.
Consequence : Current operation fails.
Recovery action : Modify for specific problem reported in cause.
Debug information : TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB”);

GENERATE_DATA_SET > TEV Internal error. > Exception MDB_ERROR raised.
—> There is a problem in the MDB. Verify messages on command line in window where
—> TEV (resp. Task Selector) was started: Are there any further messages explaining
—> the reason on the command line ?

PRINT_RESULT > System error. > UNKNOWN_PRINTER
—> The printer indicated in the environment variables VICOS_PRINTER_n was not
—> found on VICOS_PRINT_SERVER
—> Are the names set in $DBS_HOME/user_env/dbs_cshrc correct ?
—> Is a printer with this name installed and known to the node where TEV is running ?

1. QUIT_DBS_ACCESS > System error. > TDCS Error Status: TDCS_NOT_CONNECTED
2. TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB > System error. >
—> Is TDCS running ?
—> can be ignored on environments where SVF is not installed

Unix error : EACCES in LOCAL_PRINT_SERVICES SEND_FOR_PRINTING.
Permission denied PRINT_RESULT > System error. > DBS_UNIX_PROBLEM
—> can probably be ignored
—> file is put into print queue

Copyright per DIN 34

—> DBS seems to have a problem when checking the status

TEV_MAIN_MENU:TASK_TO_QUIT_TEV_CB > TEV Internal error. > QUIT_MDB_ACCESS > TEV Internal error. > Current Exception: USE_ERROR

Copyright per DIN 34

E NOTATIONAL CONVENTIONS

Notation	Description
left button **middle button** **right button**	The **Buttons** are the three buttons on the mouse. The names (**Left**, **Middle**, **Right**) refer to their position relative to the others. The names appear in bold style.
click / hold / **double click** < left | middle | right >	These indicate the action to carried out using the **left**, **middle** or **right** mouse buttons. **Click** means a short touching of the indicated mouse button. The **double click** means a fast double touching of the mouse button. **Hold** means pressing of the mouse button and releasing it after selection.
–>	A menu path is indicated by **arrow** (e.g. File–>Open). To execute a command in a menu or (series of) sub-menu hold down the indicated mouse button and drag the mouse to the right.
< **keys** | **commands** >	The names of **keys** and **commands** are shown in bold style
menu | box names	Names of **menus** and **boxes** are identified in bold and italic style.
<*variable*>	**Variables** will appear in italic style enclosed in '< >'.
`text`	Text in courier font identifies text which must be entered exactly as shown
☞	The **hand** is used when something important must be noted. The information is written in italic.
⟩	The **triangle** is used to identify a procedure with one ore more procedural steps.
▮	The **box** is used to identify procedural steps. The steps must be executed by the user.
▯	The **shadowed box** is used to identify optional procedural steps, which can be executed by the user.
•	**Bullets** list provided information, but not procedural steps.
[doc. number]	References to documents are enclosed in square brackets .

Copyright per DIN 34

F DESCRIPTION OF THE CGS TES–SAS API INTERFACE

The CGS Test Execution Software (TES) Application Programming Interface (API) comprises a set of functions as interface to a SAS. Those functions are described in detail in the following sub–sections.

Additional functions can be found in the package ADT_TES_TO_SAS_COMMAND not described in this appendix. The package contains helpful procedures which work on a lower, more detailed level.
The procedures provide functions e.g. to put parameter or message strings into a command, to set an ADU description in a command, or – vice versa – to extract parameters from a command.

The TES API comprises the following main functions:

– Connecting to and disconnecting from CGS
– Reading and handling commands
– Handling GDUs
– Handling ADUs
– Reading enditem data from CGS
– Providing enditem data to CGS
– Exchanging messages with APs
– Downloading software
– Reading the CGS time
– Reporting messages to CGS
– Store/ Retrieve of files to / from TRDB
– Start parameterless UCL Automated Procedure on TES
– Get a GDU description
– MDB translation operations

The specification of the TES_API package is presented on the following pages, a detailed description of the provided operations can be found behind it.

Most of the routines contain the following two parameters with default value:

```
Timeout    : in    Duration := P_Api_Controller.Timeout_For_Response;
Retries    : in    Natural  := P_Api_Controller.Default_Retries);
```

Description:

The Parameter Timeout defines the timeout for this operation.
The Parameter Retries defines the number of retries only in case of sending an incomplete message, if the socket is full. The timeout at all is enlarged in this case to (Retries + 1) * Timeout.

Please note that in most cases the default value should be sufficient. Therefore, a code change of existing TES_API applications will not be necessary.

```
--
-- TES_API
--
--*****************************************************************************
--ABSTRACT--
--      Provides the Application Program Interface to TES on Test Nodes for
```

Copyright per DIN 34

```
--      Specific Application Software (SAS)
--
--*------------------------------------------------------------------------

--IDENTIFICATION--
--      PROJECT NAME   : CGS
--      PRODUCT NAME   : TES
--      CI-NUMBER   : 1216 843
--      OBJECT NAME   : TES_API
--*------------------------------------------------------------------------

--CONTENTS--
--   COMPILER   : GNAT
--   LANGUAGE   : Ada
--   CHANGE HISTORY
--
--      6.3.2 at Astrium ; CGS engineering
--               creator:       Werner Tammen
--               creation date: 8.02.2007
--               CGS 6.3.2 baseline
--                  Changed the wording of the description for STORE_USER_FILE.
--      6.2.2 at Astrium ; TES engineering
--               creator:       Stephan Marz
--               creation date: 18.08.2004
--               CGS 6.2.2 baseline, PIRN 8338:
--                  Change paramater profiles:
--                  Use Ada.Strings.Unbounded.Unbounded_String instead of
--                  VICOS_DEFINITIONS.T_NAME_STRING  or VICOS_DEFINITIONS.T_NAME
--      6.0 at Astrium ; TES engineering
--               creator:       Stephan Marz
--               creation date: 15.05.03
--               CGS 6.0 baseline, PIRN 8304/A:
--                  replace MPS_DEFINITIONS.DYNAMIC_STRING by Ada.Strings.Bounded
--      6.0 at Astrium ; TES engineering
--               creator:       Stephan Marz
--               creation date: 25.02.03
--               CGS 6.0 baseline, PIRN 8304:
--                  add new function/procedure
--                      . Smt_Notification
--                      . Calculate_Command_Id
--                      . Get_User_Defined_Hk_Values
--                      . Get_Hk_Value
--                      . Set_Hk_Value
--                  add exception
--                      . Unknown_Hk_Id
--
--      5.1.1 at Astrium ; CGS engineering
--               creator:       Werner Tammen
--               creation date: 09.08.2002
--               CGS 5.1.1 baseline, PIRN 8256:
--                  Changed the wording of the description for RETRIEVE_USER_FILE.
--      5.1.1 at Astrium ; TES engineering
--               creator:       Stephan Marz
--               creation date: 13.05.2002
--               CGS 5.1.1 baseline, PIRN 8243:
--                  Change paramater profiles. Insert new parameters
--                  => Timeout [duration]
--                        for command timeout and
```

Copyright per DIN 34

```
--                 => Retries [natural]
--                       for number of retries to complete the sending
--                       in case an incomplete message is send.
--                       If the message is still incomplete after all retries,
--                       a reconnect (with data lost) will be performed.
--                 => modified parameter profile of STORE_USER_FILE and
--                    RETRIEVE_USER_FILE
--      5.1 at Astrium ; TES engineering
--               creator:       Stephan Marz
--               creation date: 18.02.02
--               CGS 5.1 baseline, PIRN 8237:
--                  Restrict ACK_CODE by new subtype Sas_Acknowledge_Code.
--      5.0 at Astrium ; TES engineering
--               creator:       Baldewein
--               creation date: 20.06.01
--               CGS 5.0 baseline, PIRN 8222:
--                  Add operations to translate SID/PATHNAME/NICKNAME.
--                  Add operations for read long float and unsigned interger values.
--                  Add operation for write long float engineering values.
--                  Add operation for get GDU from SID or PATH.
--                  This operations have a configurable TIMEOUT parameter in the
--                  CGS configuration file, called TES.API.CONTROLLER.TIMEOUT_FOR_READ
--                  Add operations for read and recieve TRDB files.
--      4.3 at DASA ; TES engineering
--               creator:       Hartmann
--               creation date: 07.01.00
--               CGS 4.3 baseline, PIRN 8096:
--                  Add operation to inform TES about ADU interruption
--
--      4.2 at DASA ; TES engineering
--               creator:       Athmann
--               creation date: 2.11.98
--               CGS 4.2 baseline, PIRN 8035:
--                  Document Use of Signal Handling instead of Polling
--
--      4.1 at DASA ; TES engineering
--               creator:       hartmann
--               creation date: 29.10.97
--               CGS 4.1 baseline, PIRN 7050/A
--
--      4.1 at DASA ; TES engineering
--               creator:       maron (on host aiv-1)
--               creation date: 08.10.97 12:00:00
--               CGS 4.1 baseline, PIRN 7050
--
--      3.0 at ERNO ; CGS engineering
--               creator:       maron (on host vicos2-2)
--               creation date: 17.03.95 12:00:00
--               Build 2 baseline, PIRN 3058
--
--      2.5 at ERNO ; VICOS development
--               creator:       athmann (on host vicos_5)
--               creation date: 03.03.93 21:02:57
--               Update acc. to COL-MBER-VIC-IRN-2047
--
--      2.4 at ERNO ; VICOS development
--               creator:       vicos (on host prime_1)
--               creation date: 18.09.92 12:12:22
```

Copyright per DIN 34

```
--               modifications according to FINAL ICD REVIEW DNs (see list)
--      2.3 at ERNO ; VICOS development
--               creator:       vicos (on host vicos_8)
--               creation date: 22.07.92 16:04:00
--               Alignment to SPADM and TN COL_MBER_ZU1_TN_0026_92
--      2.1 at ERNO ; VICOS development
--               creator:       vicos (on host vicos_4)
--               creation date: 25.06.92 11:09:36
--               INCREMENT_VERSION v1_0_0
--      1.5 at ERNO ; VICOS development
--               creator:       vicos (on host vicos_4)
--               creation date: 25.06.92 09:09:02
--               deleted old VICOS HEADER
--      1.4 at ERNO ; VICOS development
--               creator:       vicos (on host vicos_4)
--               creation date: 25.06.92 08:21:11
--               added new header

--******************************************************************************
--

with POSIX.SIGNALS;
-- type SIGNAL

with Ada.Strings.Unbounded;
-- type Unbounded_String

with Ada.Strings.Bounded;
-- for typ bounded_string

with CGS_CALENDAR;
-- type T_DATE_AND_TIME
-- type T_TIME_TYPE

with NUMERIC_TYPES;
-- type UNSIGNED_INTEGER32
-- type INTEGER32
-- type SINGLE_FLOAT
-- type DOUBLE_FLOAT

with VICOS_DEFINITIONS;
-- type T_APPLICATION_NAME
-- type T_LOG_TYPE
-- type T_LOG_TEXT_SHORT
-- type T_LOG_TEXT_LONG
-- type T_RAW_VALUE_ALTERNATIVES
-- type T_FILE_NAME

with MPS_DEFINITIONS;
-- type SID
-- type DYNAMIC_STRING
-- type T_SW_TYPE
-- type STATE_CODE

with TES_DEFINITIONS;
-- type T_APPLICATION_ID
-- type T_RETURN_STATUS
-- type T_IDENTIFIER
```

Copyright per DIN 34

```
-- type T_LINK_STATUS
-- type T_AP_MESSAGE
-- type T_ACQUISITION_STATUS
-- type T_MONITOR_STATUS

with ADT_TES_TO_SAS_COMMAND;
-- type T_COMMAND

with ADT_ADU;
-- type T_ADU

with ADT_GDU_DESCRIPTION;
-- type T_GDU_DESCRIPTION

with P_Command_Id;
-- type Command_Id;

with Cgs.Configuration.Tes.Api; use Cgs.Configuration;
-- use constant definitions for different timeouts / retries
--
-- written in CGS configuration file:
--
-- constant Timeout_For_Read   = "TES.API.CONTROLLER.TIMEOUT_FOR_READ"
--    . timeout for send a request message to TES and await an anwser ( 5.0 [s] )
-- constant Timeout_For_Send   = "TES.API.CONTROLLER.TIMEOUT_FOR_SEND"
--    . timeout for send a message to TES  ( 0.5 [s] )
-- constant Timeout_For_Response  = "TES.API.CONTROLLER.TIMEOUT_FOR_RESPONSE"
--    . timeout for send an acknowledge to TES ( 2.0 [s] )
-- constant Default_Retries       = "TES.API.CONTROLLER.DEFAULT_RETRIES"
--    . number of retries only in case of send an incomplete messages, if the socket
--      is full (3)

package TES_API is


   pragma Ident ("@(#) tes_api.ads cgs_6.3/2  02/20/07 09:41:47@(#)");

   package Api_Strings is new Ada.Strings.Bounded.Generic_Bounded_Length (255);

   type T_SMT_STATUS is (NOT_INITIALISED,STOPPED,RUNNING);

   subtype Sas_Acknowledge_Code is Integer range 0 .. 2**16-1;

   Timeout_For_Read : constant Duration :=
     Duration(Tes.Api.Controller.Timeout_For_Read) * 0.001;

   Timeout_For_Send : constant Duration :=
     Duration(Tes.Api.Controller.Timeout_For_Send) * 0.001;

   Timeout_For_Response : constant Duration :=
     Duration(Tes.Api.Controller.Timeout_For_Response) * 0.001;

   Default_Retries : constant Natural :=
     Natural(Tes.Api.Controller.Default_Retries);

   procedure CONNECT
     (SAS_NAME             : in     VICOS_DEFINITIONS.T_APPLICATION_NAME;
      CGS_PARENT           : in     VICOS_DEFINITIONS.T_APPLICATION_NAME
```

Copyright per DIN 34

```
     := VICOS_DEFINITIONS.EMPTY_NAME_STRING;
   APPLICATION_ID       :    out TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS               :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Ok_Status_When_Retry : in     Boolean := True);

-- DESCRIPTION
-- This procedure establishes a connection between a SAS and the CGS product
-- TES.
--
-- The parameter CGS_PARENT describe the logical name of TES to connect.
--
-- CGS recomment to set the parameter CGS_PARENT for testing purposes only!
-- The default will establish a better (saver) connection stategy.
--
-- (1) In the default case (CGS_PARENT = VICOS_DEFINITIONS.EMPTY_NAME_STRING),
--     the TES_API try to connect to calling TES. This information is transmitted
--     internally via environment variable $_CGS_PARENT.
-- (2) The parameter CGS_PARENT is set by user. In this case, TES_API try to
--     connect to the given TES instance.
--
-- The parameter Ok_Status_When_Retry defines whether the status
-- SUCCESS or SUCCESS_WITH_RETRIES is used for a successfull send with at
-- least one retry for all follow operations with parameter Retries.
--      Ok_Status_When_Retry = True   => SUCCESS for all successful sends
--      Ok_Status_When_Retry = False  => SUCCESS_WITH_RETRIES for send
--                                       with at least one retry
--                                    => SUCCESS for send without retry

-- Status is the error status returned by this call

-- RETURN STATES
-- SUCCESS             : the call was successfull
-- CURRENT_MODE_WRONG  : TES is in idle or error mode or it is currently not
--                       running
-- COMMUNICATION_ERROR : some communication problem exists, carefully look
--                       into your system topology table
-- SAS_UNKNOWN         : The SAS_NAME is wrong, this SAS has not been
--                       started by the CGS_PARENT
-- OTHER_ERROR           : an unexpected error occurred or the SAS has not
--                         called READ_COMMAND in time, i.e. the time-out
--                         has expired (configurable in the CGS configuration file)


procedure DISCONNECT
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS);

-- DESCRIPTION
-- This procedure releases a connection between a SAS and TES. It tries to
-- disconnect from TES.The Application_ID is the handle provide by TES in
-- the connect operation. Status is the error status returned by this call.

-- RETURN STATES
-- SUCCESS               : the call was successfull
-- NOT_CONNECTED         : the SAS is not yet connected
-- COMMUNICATION_ERROR   : some communication problem exists, carefully look
--                         into your system topology table, maybe TES died
--                         in the meantime
```

Copyright per DIN 34

```
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure READ_COMMAND
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   COMMAND        : in out ADT_TES_TO_SAS_COMMAND.T_COMMAND;
   COMMAND_ID     : in out TES_DEFINITIONS.T_IDENTIFIER;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   BLOCK          : in     BOOLEAN := FALSE);

-- DESCRIPTION
-- This procedure reads a command from TES. It needs the application id
-- returned during the connect operation in order to check that this SAS
-- is properly connected to TES.
-- The command received from TES is returned in
-- ADT_TES_TO_SAS_COMMAND.T_COMMAND. The procedures and function of package
-- ADT_TES_TO_SAS_COMMAND can be used to further process the command and
-- find out what type of caommand it is. Also, the ADT provides all
-- necessary data associated with the command. Each command also comes with
-- a COMMAND_ID which is lateron used to e.g. acknowledge the command.
-- The parameter BLOCK indicates whether the procedure shall block in the
-- call untilthere is a command from TES (if set to TRUE). In case it is
-- FALSE, the procedure will immediately return. The command is then either
-- a meaningful one or the NULL command.
--
-- It is important to notice that commands have to be acknowledged within a
-- given time interval which can be obtained from the command itself through
-- function ADT_COMMAND.COMMAND_TIMEOUT

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure ACKNOWLEDGE_COMMAND
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   COMMAND_ID     : in     TES_DEFINITIONS.T_IDENTIFIER;
   ACKNOWLEDGED   : in     BOOLEAN;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Response;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure acknowledges a command received from TES. Use
-- ADT_TES_TO_SAS_COMMAND to find out which commands need explicite
```

Copyright per DIN 34

```
-- acknowledgement.
-- It needs the application id returned during the connect operation in
-- order to check that this SAS is still connected to TES.
-- The command id as received from TES during READ_COMMAND is an input
-- parameter also.
-- If SAS wants to acknowledge the command, parameter ACKNOWLEDGED is set
-- to TRUE, else to FALSE.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- INVALID_COMMAND_ID     : The identifier is wrong, perhaps an old one is
--                          used?
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- COMMAND_NEEDS_NO_ACK   : The command needs no acknowledge
--
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure ACKNOWLEDGE_COMMAND
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   COMMAND_ID     : in     TES_DEFINITIONS.T_IDENTIFIER;
   ACKNOWLEDGED   : in     BOOLEAN;
   ACK_CODE       : in     Sas_Acknowledge_Code;
   ACK_LOT        : in     CGS_CALENDAR.CGS_DATE_AND_TIME
     := CGS_CALENDAR.NULL_DATE_AND_TIME;
   ACK_SMT        : in     CGS_CALENDAR.CGS_DATE_AND_TIME
     := CGS_CALENDAR.NULL_DATE_AND_TIME;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Response;
   Retries        : in     Natural  := Default_Retries);


-- DESCRIPTION
-- This procedure acknowledges a command received from TES. It works similar
-- to the other ACKNOWLEDGE_COMMAND procedure above but it allows the SAS to
-- specify in addition:
--  - an ACK_CODE, identifying more precisely the success or type of error
--    occurred when executing the command.
```

Copyright per DIN 34

```
--    This value is return by some UCL routines, which send commands to SAS.
--    The value range is restricted by type Sas_Acknowledge_Code.
--
--    REMARK: The complete range of Sas_Acknowledge_Code is only allowed for
--    ------  an acknowledge a GDU_REQUEST!
--            For the acknowledge to all other commands the range is restricted
--            to range 1000..Sas_Acknowledge_Code'Last. The acknowledge will
--            send to TES nevertheless in case of ACK_CODE < 1000, but the code
--            is not visible in UCL return code and an error message will be
--            generated in TES (except: ACK_CODE = 0 – no error message).
--
--  – an ACK_LOT, identifying precisely, when the command has been executed
--     (based on LOT)
--  – an ACK_SMT, identifying precisely, when the command has been executed
--     (based on SMT)
--
-- It needs the application id returned during the connect operation in
-- order to check that this SAS is still connected to TES.
-- The command id as received from TES during READ_COMMAND is an input
-- parameter also.
--
-- If SAS wants to acknowledge the command, parameter ACKNOWLEDGED is set
-- to TRUE, else to FALSE.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- INVALID_COMMAND_ID     : The identifier is wrong, perhaps an old one is
--                          used?
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- COMMAND_NEEDS_NO_ACK   : The command needs no acknowledge
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure ACKNOWLEDGE_COMMAND_WITH_STATUS
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   COMMAND_ID     : in     TES_DEFINITIONS.T_IDENTIFIER;
   SAS_STATUS     : in     TES_DEFINITIONS.T_LINK_STATUS;
   SAS_ERRORS     : in     INTEGER;
```

Copyright per DIN 34

```
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   SAS_LAST_ERROR : in     VICOS_DEFINITIONS.T_LOG_TEXT_LONG;
   Timeout        : in     Duration := Timeout_For_Response;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure is the answer to a STATUS_REQUEST command received from
-- TES. A SAS must call this procedure within the defined timeout interval,
-- otherwise TES will detect a time out.
-- The application id is the ID returned by TES during the connect operation.
-- The command id is the one returned during READ_COMMAND.
-- SAS_STATUS is the current state of the SAS. The SAS_ERRORS is the
-- accumulated number of errors detected in this SAS. It is up to the SAS
-- to decide how this counter is maintained. The SAS_LAST_ERROR is a string
-- providing more subtle information on the last error encountered.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- INVALID_COMMAND_ID     : The identifier is wrong, perhaps an old one is
--                          used?
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- COMMAND_NEEDS_NO_ACK   : The command needs no acknowledge
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure SEND_MESSAGE_TO_AP
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   AP_NAME        : in     STRING;
   MESSAGE        : in     TES_DEFINITIONS.T_AP_MESSAGE;
   MESSAGE_LENGTH : in     POSITIVE;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure sends a message to an AP's message buffer. The application
-- id is the ID returned by TES during the connect operation.The AP name is
-- the pathname of the AP the message shall be sent to. In case several
-- instances of the AP are running, TES will put the message into all input
-- buffers.
-- The message is a string to be sent to the TES AP. The length is given in
-- parameter MESSAGE_LENGTH.
```

Copyright per DIN 34

```
-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                           died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- AP_NOT_FOUND           : The AP does not exist on the testnode
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure SEND_ERROR_MESSAGE
  (APPLICATION_ID      : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ERROR_MESSAGE_SHORT : in     VICOS_DEFINITIONS.T_LOG_TEXT_SHORT;
   ERROR_MESSAGE_LONG  : in     VICOS_DEFINITIONS.T_LOG_TEXT_LONG;
   ERROR_TYPE          : in     VICOS_DEFINITIONS.T_LOG_TYPE;
   LOCAL_TIME_TAG      : in     CGS_CALENDAR.CGS_DATE_AND_TIME
     := CGS_CALENDAR.NULL_DATE_AND_TIME;
   SMT_TIME_TAG        : in     CGS_CALENDAR.CGS_DATE_AND_TIME
     := CGS_CALENDAR.NULL_DATE_AND_TIME;
   SHOW_IT             : in     BOOLEAN := TRUE;
   STATUS              :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout             : in     Duration
     := Timeout_For_Send;
   Retries             : in     Natural
     := Default_Retries);

-- DESCRIPTION
-- This procedure sends an error message to the CGS product TES. The error
-- message is always written into the CGS test result database. By default,
-- it is also shown to the user on workstations. Only if the parameter
-- SHOW_IT is set to FALSE, then it is not shown but only logged.
-- The APPLICATION_ID is the ID returned by TES during the connect operation.
-- ERROR_MESSAGE_SHORT and ERROR_MESSAGE_LONG are two text fields of the
-- message. The short text is always immediately shown on the screen of the
-- workstation (provided SHOW_IT is TRUE) whereas the long text is only
-- visible if the user double-clicks to the error message on the screen.
-- Thus short and long texts should be carefully chosen.
-- The ERROR_TYPE can be set by the SAS as needed. "WRN" for warnings, "MSG"
-- or "INFO" for messages providing just information to the user and "ERR"
-- for reports on error shall be used (refer to VICOS_DEFINITIONS for resp.
```

Copyright per DIN 34

```
-- constants).
-- The group code for the error message is set by CGS to ”SAS ”.
-- Two time_tags can be provided with each error message, being local time
-- and SMT. If the NULL_DATE_AND_TIME is given, CGS will add the actual time
-- as the time tag.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure READ_TIME
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   LOCAL_TIME     :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
   SMT            :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
   SMT_RUNNING    :    out BOOLEAN;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS);

-- DESCRIPTION
-- This procedure returns the actual local time and SMT as well as the
-- current state of the SMT, i.e. whether it is running or not.
-- The application id is the one returned by TES during the connect
-- operation.

-- RETURN STATES
-- SUCCESS   : the call was successfull
-- TSS_ERROR : the time synchronisation sw could not provide timing
--             information for unknown reasons


procedure READ_SMT_DETAILS
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   SMT              :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
   SYNC_SW_STATUS   :    out BOOLEAN;
   SYNC_SW_LOCKED   :    out BOOLEAN;
   SYNC_SW_IN_LIMIT :    out BOOLEAN;
   SMT_SERVER       :    out BOOLEAN;
   MTU_PRESENT      :    out BOOLEAN;
```

Copyright per DIN 34

```
   SMT_STATUS       :    out T_SMT_STATUS;
   SMT_OFFSET       :    out INTEGER;
   MTU_STATUS       :    out INTEGER;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS);

-- DESCRIPTION
-- This procedure returns the detailed status of the SMT.
-- The application id is the one returned by TES during the connect
-- operation.
-- SYNC_SW_STATUS returns, whether the NTP Status file, which is updated
-- every, minute by a crontab job if NTP is running, was read and has the
-- correct format
-- SYNC_SW_LOCKED returns, whether synchronisation source for the local NTP
-- process is considered as reliable.
-- SYNC_SW_IN_LIMIT returns, whether the time deviation between the NTP
-- client and its synchronisation source is in the allowed limit
-- SMT_SERVER returns, whether the local machine is SMT server (TRUE) or not
-- MTU_PRESENT returns, whether a MTU is present (TRUE) or not
-- SMT_STATUS returns the actual SMT status
-- SMT_OFFSET returns the actual SMT offset
-- MTU_STATUS returns the actual MTU status

-- RETURN STATES
-- SUCCESS   : the call was successfull
-- TSS_ERROR : the time synchronisation sw could not provide timing
--             information for unknown reasons



procedure Smt_Notification(ENABLE : in     BOOLEAN;
                           SIGNAL : in     POSIX.SIGNALS.SIGNAL;
                           STATUS :    out TES_DEFINITIONS.T_RETURN_STATUS);

-- DESCRIPTION:
-- The central Time-Server-Process (TSP) is always involved if the state
-- or value of the SMT is changed.
-- This procedure requests the TSP to send a notification ( signal ) to a
-- process, if the state of the SMT changes. The notified client can then
-- call the READ_SMT procedure to check the SMT for differences. It can
-- happen, that the TSP sends a signal allthough the SMT has not changed.
-- This should be regarded by the client.
-- The client can request the sending of more than one signal to more than
-- one thread/process.
--
-- REMEMBER:
-- The usage of the procedure requires a signal handler in the calling
-- task. The signal handler is not part of this API.
--
-- Setting ENABLE to false switches the sending off. If the TSP is not
-- able to send a signal to a process, no more notifications are send
-- to this process.
--
-- PARAMETERS:
-- ENABLE:           Enables/disables SMT notification service
-- SIGNAL:           The SIGNAL to be send in case
-- RETURN_CODE:      The return code of the request. See below.
--
-- RETURN STATES
```

Copyright per DIN 34

```
-- SUCCESS         : the call was successfull
-- COMMAND_TIMEOUT : an timeout occurred
-- TSS_ERROR       : a time synchronisation sw error occured
-- OTHER_ERROR     : a API problem occured



procedure WAIT_UNTIL
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   WAKE_UP_TIME   : in     CGS_CALENDAR.CGS_DATE_AND_TIME;
   TIME_TYPE      : in     CGS_CALENDAR.T_TIME_TYPE;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS);

-- DESCRIPTION
-- This procedure waits until a given wake up time which can be either local
-- time or SMT depending on parameter Time_Type.
-- The application id is the one returned by TES during the connect
-- operation.

-- RETURN STATES
-- SUCCESS   : the call was successfull
-- SMT_RESET : the SMT has been reset so the wake up time may not be correct.
-- TSS_ERROR : the time synchronisation sw could not provide timing
--             information for unknown reasons


------------------------------------------------------------------
-- ADU SERVICES
------------------------------------------------------

procedure ANNOUNCE_ADU_SERVICE
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure announces to CGS that the SAS is now able to handle ADUs
-- respectively ADU requests.The application id is the ID returned by TES
-- during the connect operation.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
```

Copyright per DIN 34

```
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure WITHDRAW_ADU_SERVICE
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure announces to CGS that the SAS is no longer able to handle
-- ADUs respectively ADU requests.The application id is the ID returned by
-- TES during the connect operation.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure SUPPLY_ADU
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ADU            : in     ADT_ADU.T_ADU;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Send;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure sends an ADU to TES.The application id is the ID returned
-- by TES during the connect operation.The ADU is the ADU to be sent to TES.
```

Copyright per DIN 34

```
-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.


-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure SUSPEND_ADU
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ADU_SID        : in     MPS_DEFINITIONS.SID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Send;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure indicates to TES that the ADU is from now on suspended
-- until a new ADU is sent. The application id is the ID returned
-- by TES during the connect operation.
-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
```

Copyright per DIN 34

```
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


------------------------------------------------------------------
-- GDU SERVICES
------------------------------------------------------

procedure ANNOUNCE_GDU_SERVICE
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure announces to CGS that the SAS is now able to handle
-- GDUs.The application id is the ID returned by TES during the connect
-- operation.
-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


procedure WITHDRAW_GDU_SERVICE
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure announces to TES that the SAS is no longer able to handle
-- GDUs.The application id is the ID returned by TES during the connect
-- operation.

-- Parameter Timeout defines the timeout for this operation.
```

Copyright per DIN 34

```
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred or the SAS has not
--                          called READ_COMMAND in time, i.e. the time-out
--                          has expired (configurable in the CGS configuration file)


-------------------------------------------------------------------
-- DATA PROCESSING SERVICES
-------------------------------------------------

-- SPECIAL NOTE for all read and write operation
-- --------------------------------------------

-- All read, write and translation operation have a configurable TIMEOUT
-- parameter per default. It will be set in the CGS configuration file with
-- "TES.API.CONTROLLER.TIMEOUT_FOR_READ".



procedure READ_RAW_VALUE
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME  : in     STRING;
   VALUE_ALTERNATIVE :    out VICOS_DEFINITIONS.T_RAW_VALUE_ALTERNATIVES;
   INT_VALUE         :    out NUMERIC_TYPES.INTEGER32;
   UINT_VALUE        :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   FLOAT_VALUE       :    out NUMERIC_TYPES.SINGLE_FLOAT;
   DFLOAT_VALUE      :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   BYTESTREAM_VALUE  : in out Api_Strings.Bounded_String;
   STATUS            :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure read the current raw value of an enditem from CGS. Since
-- enditems can have several raw value alternatives, the procedure returns
-- the type of raw value in parameter VALUE_ALTERNATIVE. Depending on the
-- value of this parameter, the other ..._VALUE parameter will hold the
-- actual raw value returned.  The other ones will be zeroed or empty.
```

Copyright per DIN 34

```
-- APPLICATION_ID is the ID returned by TES during the connect operation.
-- The enditem is identified through its database pathname. For simplicity
-- reasons this is a string

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- ENDITEM_NOT_ACQUIRED   : the enditem is currently not acquired, so there
--                          is no raw value
-- ENDITEM_UNKNOWN        : the enditem is unknown, so the pathname is wrong
--                          or the item is maintained on another test node.
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- TYPE_MISMATCH
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred


procedure READ_ENGINEERING_VALUE
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_Pathname  : in     STRING;
   VALUE_ALTERNATIVE :    out MPS_DEFINITIONS.T_SW_TYPE;
   INT_VALUE         :    out NUMERIC_TYPES.INTEGER32;
   UINT_VALUE        :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   FLOAT_VALUE       :    out NUMERIC_TYPES.SINGLE_FLOAT;
   DFLOAT_VALUE      :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   STATECODE_VALUE   :    out MPS_DEFINITIONS.STATE_CODE;
   BYTESTREAM_VALUE  : in out Api_Strings.Bounded_String;
   STATUS            :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure read the current engineering value of an enditem from CGS.
-- Since enditems can have several value alternatives, the procedure returns
-- the type of value in parameter VALUE_ALTERNATIVE. Depending on the value
-- of this parameter, the other ..._VALUE parameter will hold the actual
-- engineering value returned. The other ones will be zeroed or empty.
-- Note that although MPS_DEFINITIONS.T_SW_TYPE provides many more
-- alternatives only integer, unsigned_integer, float, double_float,
-- statecode or bytestream values can be returned since these correspond
```

Copyright per DIN 34

```
-- to the list of allowed CGS database measurement or SW variable enditems.
-- APPLICATION_ID is the ID returned by TES during the connect operation.
-- The enditem is identified through its database pathname. For simplicity
-- reasons this is a string

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- ENDITEM_NOT_ACQUIRED   : the enditem is currently not acquired, so there
--                          is no value
-- ENDITEM_UNKNOWN        : the enditem is unknown, so the pathname is wrong
--                          or the item is maintained on another test node.
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- TYPE_MISMATCH
-- OTHER_ERROR            : an unexpected error occurred


--   Reading Measurements

procedure READ_MEASUREMENT
  (APPLICATION_ID       : in     TES_DEFINITIONS.T_APPLICATION_ID;
   SID                  : in     MPS_DEFINITIONS.SID;
   RAW_ALTERNATIVE      :    out VICOS_DEFINITIONS.T_RAW_VALUE_ALTERNATIVES;
   RAW_INT_VALUE        :    out NUMERIC_TYPES.INTEGER32;
   RAW_UNIT_VALUE       :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   RAW_FLOAT_VALUE      :    out NUMERIC_TYPES.SINGLE_FLOAT;
   RAW_DFLOAT_VALUE     :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   RAW_BYTESTREAM_VALUE : in out Api_Strings.Bounded_String;
   VALUE_ALTERNATIVE    :    out MPS_DEFINITIONS.T_SW_TYPE;
   INTEGER_VALUE        :    out NUMERIC_TYPES.INTEGER32;
   UINTEGER_VALUE       :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   FLOAT_VALUE          :    out NUMERIC_TYPES.SINGLE_FLOAT;
   DFLOAT_VALUE         :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   STATECODE_VALUE      :    out MPS_DEFINITIONS.STATE_CODE;
   BYTESTREAM_VALUE     : in out Api_Strings.Bounded_String;
   TIME_TAG             :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
   ENDITEM_ACQ_STATUS   :    out TES_DEFINITIONS.T_ACQUISITION_STATUS;
   MONITORING_STATUS    :    out TES_DEFINITIONS.T_MONITOR_STATUS;
   STATUS               :    out TES_DEFINITIONS.T_RETURN_STATUS;
```

Copyright per DIN 34

```
   Timeout              : in     Duration
     := Timeout_For_Read;
   Retries              : in     Natural
     := Default_Retries);

-- DESCRIPTION
-- This procedure read the current raw and engineering value of an enditem from CGS.
-- Since enditems can have several value alternatives, the procedure returns
-- the type of value in parameter RAW_ALTERNATIVE and VALUE_ALTERNATIVE.
-- Depending on the value
-- of this parameter, the other ..._VALUE parameter will hold the actual
-- engineering value returned. The other ones will be zeroed or empty.
-- Note that although MPS_DEFINITIONS.T_SW_TYPE provides many more
-- alternatives only integer, unsigned_integer, float, double_float,
-- statecode or bytestream values can be returned since these correspond
-- to the list of allowed CGS database measurement or SW variable enditems.
-- APPLICATION_ID is the ID returned by TES during the connect operation.
-- The enditem is identified through its database SID.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- Return status values:
--
--  SUCCESS
--  SUCCESS_WITH_RETRY
--  OTHER_ERROR
--  ENDITEM_UNKNOWN
--  CURRENT_MODE_WRONG
--  TYPE_MISMATCH
--  COMMUNICATION_ERROR
--  COMMAND_TIMEOUT
--  NOT_SEND_IN_TIME
--  RECONNECTED
--  NO_RESPONSE_IN_TIME
--  INVALID_APPLICATION_ID

procedure READ_MEASUREMENT
  (APPLICATION_ID        : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME      : in     STRING;
   RAW_ALTERNATIVE       :    out VICOS_DEFINITIONS.T_RAW_VALUE_ALTERNATIVES;
   RAW_INT_VALUE         :    out NUMERIC_TYPES.INTEGER32;
   RAW_UNIT_VALUE        :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   RAW_FLOAT_VALUE       :    out NUMERIC_TYPES.SINGLE_FLOAT;
   RAW_DFLOAT_VALUE      :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   RAW_BYTESTREAM_VALUE  : in out Api_Strings.Bounded_String;
   VALUE_ALTERNATIVE     :    out MPS_DEFINITIONS.T_SW_TYPE;
   INTEGER_VALUE         :    out NUMERIC_TYPES.INTEGER32;
   UINTEGER_VALUE        :    out NUMERIC_TYPES.UNSIGNED_INTEGER32;
   FLOAT_VALUE           :    out NUMERIC_TYPES.SINGLE_FLOAT;
   DFLOAT_VALUE          :    out NUMERIC_TYPES.DOUBLE_FLOAT;
   STATECODE_VALUE       :    out MPS_DEFINITIONS.STATE_CODE;
   BYTESTREAM_VALUE      : in out Api_Strings.Bounded_String;
   TIME_TAG              :    out CGS_CALENDAR.CGS_DATE_AND_TIME;
   ENDITEM_ACQ_STATUS    :    out TES_DEFINITIONS.T_ACQUISITION_STATUS;
   MONITORING_STATUS     :    out TES_DEFINITIONS.T_MONITOR_STATUS;
```

Copyright per DIN 34

```
   STATUS               :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout              : in     Duration := Timeout_For_Read;
   Retries              : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure read the current raw and engineering value of an enditem from CGS.
-- Since enditems can have several value alternatives, the procedure returns
-- the type of value in parameter RAW_ALTERNATIVE and VALUE_ALTERNATIVE.
-- Depending on the value
-- of this parameter, the other ..._VALUE parameter will hold the actual
-- engineering value returned. The other ones will be zeroed or empty.
-- Note that although MPS_DEFINITIONS.T_SW_TYPE provides many more
-- alternatives only integer, unsigned_integer, float, double_float,
-- statecode or bytestream values can be returned since these correspond
-- to the list of allowed CGS database measurement or SW variable enditems.
-- APPLICATION_ID is the ID returned by TES during the connect operation.
-- The enditem is identified through its database pathname. For simplicity
-- reasons this is a string.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- Return status values:
--
--  SUCCESS
--  SUCCESS_WITH_RETRY
--  OTHER_ERROR
--  ENDITEM_UNKNOWN
--  CURRENT_MODE_WRONG
--  TYPE_MISMATCH
--  COMMUNICATION_ERROR
--  COMMAND_TIMEOUT
--  NOT_SEND_IN_TIME
--  RECONNECTED
--  NO_RESPONSE_IN_TIME
--  INVALID_APPLICATION_ID

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
   VALUE            : in     NUMERIC_TYPES.INTEGER32;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
   VALUE            : in     NUMERIC_TYPES.UNSIGNED_INTEGER32;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
```

Copyright per DIN 34

```
   VALUE            : in     NUMERIC_TYPES.SINGLE_FLOAT;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
   VALUE            : in     NUMERIC_TYPES.DOUBLE_FLOAT;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
   VALUE            : in     MPS_DEFINITIONS.STATE_CODE;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

procedure WRITE_ENGINEERING_VALUE
  (APPLICATION_ID   : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME : in     STRING;
   VALUE            : in     Api_Strings.Bounded_String;
   STATUS           :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure writes an engineering value of an enditem of type
-- EGSE_...SW_VARIABLE.
-- The application id is the ID returned by TES during the connect
-- operation.
-- The pathname is the pathname of the enditem to be written. For
-- simplicity reasons this is a string.
-- The procedure comes in overloaded versions for the different types of
-- values to be written.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
--                          not running
-- COMMUNICATION_ERROR    : some communication problem exists, carefully
--                          look into your system topology table, maybe TES
--                          died in the meantime
-- INVALID_APPLICATION_ID : The identifier is wrong, perhaps an old one is
--                          used?
-- ENDITEM_UNKNOWN        : the enditem is unknown, so the pathname is wrong
--                          or the item is maintained on another test node.
```

Copyright per DIN 34

```
-- NOT_A_SW_VALUE         : the enditem exists but is not a SW variable
-- TYPE_MISMATCH          : the enditem's type defined in the database and
--                          the data supplied do not match.
-- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached
-- OTHER_ERROR            : an unexpected error occurred

procedure STORE_USER_FILE
  (FILE_NAME    : in     STRING;
   FILE_TYPE    : in     STRING := "ANY     ";
   SESSION_NAME :    out Ada.Strings.Unbounded.Unbounded_String;
   STATUS       :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout      : in     Duration := Timeout_For_Read;
   Retries      : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This operation will store the specified file into the TRDB.
-- Note: no session has to be specified. The file is stored beneath the
--       currently opened session.
--       The copy operation into the TRDB filesystem is decoupled from the
--       checks done to verify e.g. that the file has not already been transferred
--       into the TRDB, or whether the file is accessible on the DB server,
--       in order not to block the calling application.
--       This means a status OK might be returned although the file has not yet
--       been copied into the TRDB. The following operation RETRIEVE_USER_FILE
--       can be used to verify this.
-- FILE_NAME   : Full location of the file to archive.
-- FILE_TYPE   : the type of user file.
-- SESSION_NAME: in case of any errors an empty string will be returned,
--               otherwise the name of the session.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES
-- SUCCESS                      : the call was successfull
-- DBS_CONNECTION_PROBLEM       : any problems with the DBS
-- COMMAND_TIMEOUT              : Communication Time out occured
-- FILE_NOT_FOUND_ON_LOCAL_NODE : the file to be stored is not visible on the
--                                local node
-- FILE_NOT_FOUND_ON_SERVER_NODE: the file to be stored is not visible on the
--                                database server
-- FILE_ALREADY_EXISTS          : file is already known within the TRDB for
--                                this session
-- NO_SPACE_LEFT_ON_DEVICE      : there is no more space left to copy the file
--                                into the TRDB
-- NOT_SEND_IN_TIME             : This command is not send, the given timeout reached
-- RECONNECTED                  : This command is not send (incomplete message), a
--                                reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME          : This command was send, but no response was received
--                                before the given timeout was reached
-- OTHER_ERROR                  : an unexpected error occurred
```

Copyright per DIN 34

```
procedure RETRIEVE_USER_FILE
  (SESSION_NAME : in     STRING;
   FILE_NAME    : in     STRING;
   LOCATION     :    out Ada.Strings.Unbounded.Unbounded_String;
   STATUS       :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout      : in     Duration := Timeout_For_Read;
   Retries      : in     Natural  := Default_Retries);

-- DESCRIPTION
-- This procedure retrieves a filename from the TRDB. Note: the file will be not be copied,
-- only the complete path of the file in the TRDB will be returned.

-- SESSION_NAME  : The session name.
--                 No null value is accepted.
-- FILE_NAME     : User file name of a previously stored file.
--                 Note: the file name must only contain the basename of
--                 the file.
-- LOCATION      : The TRDB location (full pathname) of the file to
--                 retrieve. The filename will be composed using the names
--                 of the TRDB directory, the session and the specified
--                 file.
-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.


-- RETURN STATES

-- SUCCESS                      : the call was successfull
-- SUCCESS_WITH_RETRY           : the call was successfull, but with retries,
--                                (depend on parameter Ok_Status_When_Retry in
--                                CONNECT operation)
-- DBS_CONNECTION_PROBLEM       : any problems with the DBS
-- COMMAND_TIMEOUT              : Communication Time out occured
-- NOT_SEND_IN_TIME             : This command is not send, the given timeout reached
-- RECONNECTED                  : This command is not send (incomplete message), a
--                                reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME          : This command was send, but no response was received
--                                before the given timeout was reached
-- INVALID_SESSION_NAME         : the name of the session is not known within
--                                the TRDB
-- SESSION_IS_ARCHIVED          : the file to be retrieved is currently stored on
--                                the final archive medium – the session has to be
--                                retrieved first
-- FILE_NOT_FOUND_ON_SERVER_NODE: the file to be retrieved is unknown in the TRDB
-- OTHER_ERROR                  : an unexpected error occurred



--   Translation SID – Pathname/Nickname
-- The translation functions have no RETURN STATES, in case of errors or
-- any problems NULL_PATHNAME or NULL_SID returns. The error will be reported.

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
```

Copyright per DIN 34

```
-- in this case to ( Retries + 1 ) * Timeout.


function PATHNAME_TO_SID
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME  : in     STRING;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries)
  return MPS_DEFINITIONS.SID;

function NICKNAME_TO_SID
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   NICKNAME          : in     STRING;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries)
  return MPS_DEFINITIONS.SID;

function GET_PATHNAME
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   SID               : in     MPS_DEFINITIONS.SID;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries)
  return STRING;

function GET_NICKNAME
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   ENDITEM_PATHNAME  : in     STRING;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries)
  return STRING;

function GET_NICKNAME
  (APPLICATION_ID    : in     TES_DEFINITIONS.T_APPLICATION_ID;
   SID               : in     MPS_DEFINITIONS.SID;
   Timeout           : in     Duration := Timeout_For_Read;
   Retries           : in     Natural  := Default_Retries)
  return STRING;

--

-- The GET_GDU functions have no RETURN STATES, in case of errors or
-- any problems the alternative undefined will returns.

function GET_GDU
  (ENDITEM_PATHNAME :        STRING;
   Timeout          : in     Duration := Timeout_For_Read;
   Retries          : in     Natural  := Default_Retries)
  return ADT_GDU_DESCRIPTION.T_GDU_DESCRIPTION;

function GET_GDU
  (SID     :        MPS_DEFINITIONS.SID;
   Timeout : in     Duration := Timeout_For_Read;
   Retries : in     Natural  := Default_Retries)
  return ADT_GDU_DESCRIPTION.T_GDU_DESCRIPTION;

-- Returns a GDU description. This GDU can be used for constructing
-- Upload TCs.
-- Parameter Timeout defines the timeout for this operation.
```

Copyright per DIN 34

```
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- EXCEPTIONS
-- GDU_TYPE_MISMATCH :
--   the ENDITEM_PATHNAME is not a GDU end item (EGSE_PREDEFINED_TC, EGSE_PUS_TC, etc)


--   Start AP without parameter

procedure START_AP
  (APPLICATION_ID : in     TES_DEFINITIONS.T_APPLICATION_ID;
   AP_NAME        : in     STRING;
   STATUS         :    out TES_DEFINITIONS.T_RETURN_STATUS;
   Timeout        : in     Duration := Timeout_For_Read;
   Retries        : in     Natural  := Default_Retries);

-- Parameter Timeout defines the timeout for this operation.
-- Parameter Retries defines the number of retries only in case of send an
-- incomplete messages, if the socket is full. The timeout at all is enlarged
-- in this case to ( Retries + 1 ) * Timeout.

-- RETURN STATES

-- SUCCESS                  : the call was successfull
-- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
--                          (depend on parameter Ok_Status_When_Retry in
--                           CONNECT operation)
-- ENDITEM_UNKNOWN          : the enditem is unknown, so the pathname is wrong
--                              or the item is maintained on another test node.
-- CURRENT_MODE_WRONG       : TES is in idle or error mode or it is currently not
--                              running
-- COMMAND_TIMEOUT          : Communication Time out occured (configurable in the
--                              CGS configuration file)
-- OTHER_ERROR              : an unexpected error occurred
-- INVALID_PARAMETER
-- AP_NOT_STOPPED
-- TOO_MANY_APS
-- AP_NOT_COMPILED
-- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
-- RECONNECTED            : This command is not send (incomplete message), a
--                          reconnect was performed (data lost)
-- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
--                          before the given timeout was reached


--------------------
-- telecommand id --
--------------------

function Calculate_Command_Id return P_Command_Id.T_Command_Id;

-- This function use the P_Command_Id packet to calculate a unique
-- tele command identifier. The P_Command_Id.T_Command_Id contains
-- the ip address, on which this call is used, a time information
-- and the process id of the calling process.
-- For detailed information about the structure see in P_Command_Id.
```

Copyright per DIN 34

```
   ----------------------------
   -- User defined HK values --
   ----------------------------

   function Get_User_Defined_Hk_Values return Tes_Definitions.Hk_Value_Array;

   -- This function returns the user defined HK Values. This values may be
   -- set via TES_API only. They are to be defined in the CGS configuration
   -- file ($CGS_HOME/etc/cgs_configuration.xml) under group Housekeeping.
   -- For the first implementation statecode values are allowed only.


   function Get_Hk_Value
     (Id      : Integer;
      Timeout : Duration := Timeout_For_Read;
      Retries : Natural  := Default_Retries)
     return Mps_Definitions.State_Code;

   -- This function returns the state code value of user defined HK Values for a
   -- special Id. This values may be set via TES_API only.
   -- In case of error an empty state code is returned.
   -- exception: Unknown_Hk_Id

   procedure Set_Hk_Value
     (Value   :     Mps_Definitions.State_Code;
      Id      :     Integer;
      Timeout :     Duration := Timeout_For_Read;
      Retries :     Natural  := Default_Retries;
      Status  : out Tes_Definitions.T_Return_Status);

   -- This function set the state code value of user defined HK Values for a
   -- special Id.
   -- RETURN STATES
   -- SUCCESS                : the call was successfull
   -- SUCCESS_WITH_RETRY     : the call was successfull, but with retries,
   --                          (depend on parameter Ok_Status_When_Retry in
   --                           CONNECT operation)
   -- CURRENT_MODE_WRONG     : TES is in idle or error mode or it is currently
   --                          not running
   -- COMMUNICATION_ERROR    : some communication problem exists, carefully
   --                          look into your system topology table, maybe TES
   --                          died in the meantime
   -- COMMAND_TIMEOUT        : The acknowledge comes too late, sorry...
   -- NOT_SEND_IN_TIME       : This command is not send, the given timeout reached
   -- RECONNECTED            : This command is not send (incomplete message), a
   --                          reconnect was performed (data lost)
   -- NO_RESPONSE_IN_TIME    : This command was send, but no response was received
   --                          before the given timeout was reached
   -- OTHER_ERROR            : an unexpected error occurred

   -- exception: Unknown_Hk_Id

   Unknown_Hk_Id: exception;
   -- exception in case of none user defined Hk value.

end TES_API;
```

Copyright per DIN 34

F-1 Connecting to and Disconnecting from CGS

Before a SAS can start communication with CGS it has to connect to CGS first. For SAS running on an HP computer, the CONNECT operation will connect the SAS to the local instance of TES on the same HP the SAS is running on. For SAS running on a SUN, the host name of the parent TES has to be provided as a parameter to the connect operation.

The mechanism to connect to CGS is the operation

CONNECT

from the TES_API package.

CONNECT has to be called fast after the SAS is started since TES has a built–in time–out mechanism which will reject SAS CONNECT requests after the elapsed time. The time interval between calling the UCL system library call to start the SAS and the SAS then calling CONNECT via the TES API is configurable through the TES config file.

If the polling rate for incoming commands shall be adjusted, then the operation SET_POLLING_INTERVAL has to be called before the CONNECT operation (see above).

Inside the SAS, the CONNECT operation can be called at any place before and/or after any internal initialisation or processing which has to be performed.

The parameters of this operation have the following meaning:

–The SAS_NAME is the name of the executable image of the SAS. This is required since TES needs to correlate the name of an application started through a UCL library command with the ones trying to connect. It is of type T_APPLICATION_NAME from the package VICOS_DEFINITIONS. This parameter has to be assigned carefully (and correctly !) since otherwise the API will return with error code 99 (OTHER_ERROR). A possible assignment looks as follows:
SAS_NAME: constant VICOS_DEFINITIONS.T_APPLICATION_NAME :=(”SAS_TMTC ”,8);
The T_APPLICATION_NAME is a record with two components, one being the name of the SAS (a twenty character string, filled with blanks, the SAS name being filled in left justified) and a length field(giving the number of valid characters for the SAS name, excluding the blanks!).

–The CGS_PARENT is the application name of the parent TES, e.g. ”TES_01”, as defined in the System Topology Table. The CGS_PARENT denotes the process logical name where the SAS shall connect to. This parameter must only be supplied in case the SAS runs on a node different from the local node. i.e. only, if the TES to connect to runs on a different node than the SAS itself.
A common mechanism to transfer the CGS_PARENT to the SAS is via the start parameter the user can give in the LOAD_APPLICATION command. The SAS can read these parameter as command line parameter and use the value given for CGS_PARENT in the CONNECT procedure.

–The Application_ID is returned by TES. It is the unique handle to be used in all future calls to the TES–API for this specific instance of TES.

–STATUS is the error status returned by this call.

Copyright per DIN 34

–The parameter Ok_Status_When_Retry defines whether the status SUCCESS or SUCCESS_WITH_RETRIES is used for a successfull send with at least one retry for all follow operations with parameter Retries.

Ok_Status_When_Retry = True => SUCCESS for all successful sends
Ok_Status_When_Retry = False => SUCCESS_WITH_RETRIES for send with at least one retry
=> SUCCESS for send without retry

After all processing within the SAS has finished, the SAS has to disconnect from CGS. This is done using the procedure

DISCONNECT

from the TES_API package. After this procedure has been called, no further communication is possible with CGS and the SAS should terminate itself. Usually, this is done in V2 by calling the UNIX _exit system service as the last executable statement. (Because not all tasks inside the TES_API can be terminated, this construct has to be used, otherwise the process could not stop due to the fact that the Ada run–time system would assume that some tasks are still active and would not allow termination). In V3 this will be changed TBD.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–STATUS is the error status returned by this call.

F-2 Reading and Handling Commands

In this section the general mechanisms of handling commands from CGS will be explained. More information can be found in the subsequent sections where the processing of certain data is described in more detail.

A SAS has to react according to commands received from TES, normally. The mechanism to accept such commands is the procedure

READ_COMMAND

from the TES_API package. This procedure delivers exactly one command to the SAS in a blocking or non–blocking mode.

The parameters of this operation have the following meaning:

–APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–COMMAND is the command read (see below for an explanation of how to deal with this command.

–COMMAND_ID is a unique identifier for this command to be used in all subsequent replies to CGS with respect to this command, e.g. an acknowledge.

Copyright per DIN 34

–STATUS is the error status returned by this call.

–BLOCK is indicating whether the procedure shall block until a command is sent from CGS or not. In case the parameter is set to false, the procedure will not block and will always return, either with a command or with no command. In case it is set to true, then the procedure will only return after a command has been sent to the SAS.

Types of Commands

The following types of commands can be sent from CGS to a SAS:

–UNDEFINED

This type of command indicates some error in the communication link to CGS and this command should normally not appear. In this case the SAS should send an error message to CGS.

–INIT_APPLICATION

This command is sent whenever the CGS user wants to initialise a SAS through a call to the corresponding UCL system library procedure (INIT_APPLICATION). Together with the command, an initialisation command (string) can be passed to the SAS and it is up to the SAS to perform the proper initialisation then. CGS does not check that a SAS has been initialised nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit initialisation or not.

–START_APPLICATION

This command is sent whenever the CGS user wants to start the SAS processing internally through a call to the corresponding UCL system library procedure (START_APPLICATION). This does not mean starting a SAS as an operating system process but it means internal starting of the SAS after it has been stopped, e.g. by a RESET command.. CGS does not check that a SAS has been started nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit starting or not.

–RESET_APPLICATION

This command is sent whenever the CGS user wants to reset the SAS processing internally through a call to the corresponding UCL system library procedure (RESET_APPLICATION). This does not mean resetting a SAS as an operating system process but it means internal resetting of the SAS, e.g. after internal errors have been detected. CGS does not check that a SAS has been reset nor does it require it. Consequently, it is up to the user or the SAS programmer to define whether a SAS needs explicit resetting or not.

–STATUS_REQUEST

This command is sent whenever the CGS user wants to read the SAS status through a call to the corresponding UCL system library procedure (GET_APPLICATION_STATUS). The SAS has to respond to this command by supplying the current status (see below)

–AP_MESSAGE

This command indicates a message has been sent from an automated procedure to the SAS through the UCL system library procedure SEND_MESSAGE_TO_APPLICATION. The SAS can extract the message from the command and process it.

–ADU_REQUEST

This command indicates to the SAS that CGS wants a certain ADU to be dispatched from now on or that a previous request for dispatching an ADU shall now be cancelled. The corresponding ADU description as stored in the mission database is passed to the SAS together with this command.

Copyright per DIN 34

–SUPPLY_ADU
This command contains the ADU supplied by CGS, as previously requested by this SAS. ADU re–routing to SASes will be implemented in CGS V3 only.

–GDU_REQUEST
This command indicates to the SAS that a certain telecommand or stimulus shall be generated to the unit under test (or the test system itself). The GDU supplied as part of the command contains all necessary information in order for the SAS to generate the telecommand/stimulus to the corresponding front end hardware.

–SW_DOWNLOAD
This command indicates to the SAS that some Software replaceable unit from the mission database has to be downloaded into the unit under test or some front end equipment.

–UNLOAD_APPLICATION
This command is sent to the SAS as a result of the UCL system library procedure UNLOAD_APPLICATION. The SAS shall then disconnect from TES, stop its internal processing and terminate as an operating system process.

–NO_COMMAND
This ”command” is returned by the procedure READ_COMMAND whenever there is no command to read!

Command Acknowledgement

Commands have to be acknowledged by the SAS in a special way. For this purpose, the procedure

ACKNOWLEDGE_COMMAND

from the TES_API package has to be called.

As can be seen in the specification, three versions of this procedure exist for the following purposes:
– reply to the STATUS_REQUEST command
– simple acknowledge or advanced acknowledge of all other commands

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–COMMAND_ID is the identifier for this command returned by procedure READ_COMMAND.

–ACKNOWLEDGED is the boolean indicating to CGS whether the SAS acknowledges the command (TRUE) or not (FALSE).

–ACK_CODE identify more precisely the success or type of error occurred when executing the command. The valid range is 1000 .. 5000. **Be careful**: This code is routed to UCL/HLCL level instead of Ground_Library.SUCCESS.

–ACK_LOT identify precisely when the command has been executed (based on LOT).

Copyright per DIN 34

–ACK_SMT identify precisely when the command has been executed (based on SMT).

The ACK_CODE, the ACK_LOT and the ACK_SMT will be archived in CGS (depend on archiving *on* or *off*) and are accessible via *Ground_Library.Issue* command.

The CGS time format, provided by type *Cgs_Calendar.Cgs_Date_And_Time*, is now precise up to 1 nano second.

–SAS_STATUS is the current status of the SAS. It is and Ada enumeration type with the following alternatives:

RESET	a RESET_APPLICATION command has reset the SAS
INITIALISED	an INIT_APPLICATION command has initialised the SAS
RUNNING	a START_APPLICATION command has internally started processing, the SAS is running normally
ABORTED	the SAS has internally detected an error and has stopped processing, reset. re–initialisation or re–starting is needed possibly

–SAS_ERRORS is a simple counter value in which the SAS can return the cumulative number of errors internally detected. This value is not checked by CGS.

–SAS_LAST_ERROR is the text of the last error message or a descriptive text of the last error detected by the SAS.

–STATUS is the error status returned by this call.

How a specific type of command has to be acknowledged can be seen from the following list:

–UNDEFINED
This type of command needs no acknowledgement at all.

–INIT_APPLICATION
This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–START_APPLICATION
This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

Copyright per DIN 34

–RESET_APPLICATION

This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–STATUS_REQUEST

This command has to be acknowledges by calling alternative 1 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details. All relevant SAS status data have to be provided to CGS.

–AP_MESSAGE

This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–ADU_REQUEST

This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–SUPPLY_ADU

This type of command needs no acknowledgement at all.

–GDU_REQUEST

This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–SW_DOWNLOAD

This command has to be acknowledges by calling alternative 3 of the ACKNOWLEDGE_COMMAND procedure from the TES_API within a predefined time interval. The time–out interval period can be obtained from the command itself. Usually, all commands sent from TES to a SAS have the same time–out interval which is configurable through the TES configuration file on the relevant test node. See your CGS system administrator for more details.

–UNLOAD_APPLICATION

This type of command needs no acknowledgement at all.

Copyright per DIN 34

–NO_COMMAND
This type of command needs no acknowledgement at all.

Command parsing

The commands returned from CGS by procedure READ_COMMAND are private types in the Ada implementation sense. As such, the programmer can only parse them using the package ADT_TES_TO_SAS_COMMAND which provides the corresponding type definition in order to create Ada variables of type T_COMMAND plus all procedures to extract the relevant information from the command.

The following information is provided in a command from CGS to SAS:

–COMMAND_ALTERNATIVE
this is an enumeration type with the following possible values:
UNDEFINED
INIT_APPLICATION
START_APPLICATION
RESET_APPLICATION
STATUS_REQUEST
AP_MESSAGE
ADU_REQUEST
SUPPLY_ADU
GDU_REQUEST
SW_DOWNLOAD
UNLOAD_APPLICATION
NO_COMMAND
The list of possible values exactly corresponds to the set of commands which can be sent from CGS to a SAS.

–COMMAND_TIMEOUT
A positive number indicating in milliseconds the allowed time interval between the reception of the command and the acknowledge to CGS.

–STRING_LENGTH
In case an INIT_APPLICATION or AP_MESSAGE command is received this figure indicates the number of characters associated with the command as the message string.

–INIT_PARAMS
In case of an INIT_APPLICATION command, the initialisation string provided to the SAS through the UCL system library procedure. The length of this string can be obtained separately

–MESSAGE
In case of an AP_MESSAGE command, the message text sent from the UCL system library procedure to the SAS. The length of this string can be obtained separately

–ADU_DESCRIPTION
In case of an ADU_REQUEST command, the ADU description of the ADU to be supplied later by this SAS.

Copyright per DIN 34

–ADU
In case of a SUPPLY_ADU command, the ADU routed by CGS.

–GDU
In case of a GDU_REQUEST command, the GDU to be processed by this SAS, i.e. forwarded to the unit under test in the form of a telecommand or stimulus

–DATABASE_SW_REPLACEABLE_UNIT
In case of a SW_DOWNLOAD command, the SW to be downloaded into the unit under test or front end equipment.

Ada Command Handling Example

In the following, an example piece of Ada code is given which shows the general logic to be applied when parsing and handling commands received from CGS:

```
with   TES_DEFINITIONS;                 -- This package provides some common types
with   TES_API;                         -- this package provides the TES_API operations !
with   ADT_TES_TO_SAS_COMMAND;  -- this package provides a high level interface for
...                                     --  handling the commands sent from TES to SAS
with   Ada.Text_IO;

procedure MAIN is                       -- here the main program starts
       CMD: ADT_TES_TO_SAS_COMMAND.T_COMMAND;    -- the variable for the command
       READ_STATUS: TES_DEFINITIONS.T_RETURN_STATUS;
                                                 -- the return status of API operations
       procedure SEND_ACK_TO_CGS(...)
           is   -- a general procedure to acknowledge commands
...
       end SEND_ACK_TO_CGS;
...
begin
...                    -- do some initialisations
       loop            -- the main program loop starts here
...                    -- do some more initialisations
           TES_API.READ_COMMAND
               (...,       -- wait for a command and block
               COMMAND => CMD,
               ...,
               STATUS => READ_STATUS,
               BLOCK => TRUE);
           if READ_STATUS /= TES_DEFINITIONS.SUCCESS then -- check if READ_COMMAND was successful
               Ada.Text_IO.PUT_LINE ( "Error during reading command from TES");
           else
                   -- now parse the command using Ada CASE
               case ADT_TES_TO_SAS_COMMAND.COMMAND_ALTERNATIVE(COMMAND_FROM_TES) is
                   when ADT_TES_TO_SAS_COMMAND.INIT_APPLICATION =>
                       -- the INIT_APPLICATION_COMMAND
                       Ada.Text_IO.PUT_LINE("INIT_APPLICATION Command");
                       -- process the INIT_APPLICATION command
                       SEND_ACK_TO_CGS(COMMAND_ID);   -- acknowledge the command
                   when ADT_TES_TO_SAS_COMMAND.START_APPLICATION=>
```

Copyright per DIN 34

```
                    -- the START_APPLICATION_COMMAND
                    Ada.Text_IO.PUT_LINE("START_APPLICATION command");
                    SEND_ACK_TO_CGS(COMMAND_ID);
...                 -- more types of commands
                  when others =>
                    null;
              end case;
          end if              -- the check for a valid READ_COMMAND
      end loop                -- the main program loop ends here
end MAIN;                     -- here the main program ends
```

Copyright per DIN 34

F-3 Handling GDUs

GDUs are sent from CGS to the SAS using the GDU_REQUEST command. Before that happens, the SAS has to indicate to CGS though that it is able to handle GDUs. For this purpose, the procedure

ANNOUNCE_GDU_SERVICE

from the TES_API has to be called.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–STATUS is the error status returned by this call.

After this operation has been called, all GDUs meant for this SAS will be sent to it by CGS whenever necessary, i.e. when a UCL system library command ISSUE is encountered or when a monitoring exception occurs.

As soon as the SAS is no longer in a position to process GDUs, it can call the operation

WITHDRAW_GDU_SERVICE

from the TES_API.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–STATUS is the error status returned by this call.

After this operation has been called, CGS will refuse to send those GDUs meant for this SAS and will inform the user accordingly through appropriate error messages.

A SAS may temporarily change over its local status (ANNOUNCE_GDU_SERVICE, WITHDRAW_GDU_SERVICE, ...) at any time and more than once during a ’session’ with CGS.

When a GDU_REQUEST command is then read from CGS, the GDU is obtained from that command using the corresponding function in the ADT_TES_TO_SAS_COMMAND. This GDU can be processed internally in the SAS, sent to the corresponding front end equipment and from there to the unit under test. At any point in time the SAS is free to send the acknowledge for this command back to CGS.

It is important to note here that it is the responsibility of the SAS programmer/designer to decide, at which point in time the acknowledge for a GDU will be given to CGS. The two extreme cases are:

Copyright per DIN 34

–The acknowledge is given immediately after the command has been read and it has been decided that reading the command was successful, or

–The command is read, the stimulus is sent to the unit under test and only after an expected reaction has been seen from the unit under test or the front end equipment then the acknowledge is given to CGS

Whatever the SAS designer/programmer decides he/she must be aware of the fact that CGS checks for the predefined timeout interval when sending out a GDU to a SAS, i.e. before that interval expires the ACK/NACK must have been received by CGS.

GDU Processing

Similar to the commands sent from CGS to SAS, the GDUs are also private types, encapsulated in an Ada package. This package provides the type definition for the T_GDU plus all operations to read the information needed.

The following information is part of the GDU and available to the SAS through proper use of the ADT_GDU:

–ID
The database short identifier (SID) of this GDU_(DESCRIPTION). The GDU_DESCRIPTION is the database contents describing the GDU received from CGS.

–ALTERNATIVE
There are four GDU alternatives defined currently in CGS:

ANALOG_STIMULUS	a stimulus with an analogue parameter, e.g. SET_TEMPERATURE(VALUE) where SET_TEMPERATURE denotes the stimulus and VALUE is the actual value.
DISCRETE_STIMULUS	a fully defined stimulus, e.g. POWER_ON
PREDEFINED_TC	a CCSDS telecommand
PUS_TC	a CCSDS / PUS telecommand (PacketUtilizationStandart)
UNDEFINED	an undefined type (not used)

–VALUE_OF
In case of an ANALOG_STIMULUS, the actual value of the parameter. In case of a DISCRETE_STIMULUS the value is an integer derived from the STATECODE value supplied as a parameter on UCL level

–PRIORITY
This indicates the priority with which the GDU has been handled inside CGS. It should be good practice to handle GDUs with similar priority inside the SAS also.
CGS uses three levels of priority:
NORMAL: the lowest priority
HIGH: all GDUs with high priority interrupt the sending of NORMAL priority GDUs
EMERGENCY: these GDUs are generated as a result of monitoring exceptions and have highest priority.

Copyright per DIN 34

–TIME_TAG

The point in time when this GDU shall really be sent to the unit under test.
IMPORTANT : If the GDU contains a CCSDS packet, the time tag field of the GDU can always be ignored by the SAS since the time tag actually supplied when sending the GDU/CCSDS packet is put into the secondary header of the CCSDS packet.

–SEQUENCE_COUNT

In case of CCSDS predefined TCs, the sequence count of the CCSDS packet being contained in this GDU

–RETRIES

The number of retries defined for this GDU in the database

Copyright per DIN 34

–PHYSICAL_ADDRESS

The low level device specific information needed by the SAS to really issue the GDU. This information is stored in the configuration database and passed to the SAS by CGS. CGS does not interpret it. It is up to the SAS to decide, how the information in the physical address field should be used. It is, however, good practice to re–use as much from the information contained herein, in order to ensure consistency enforced by the use of the configuration database also in the SAS area and in order to simplify reconfiguration of front end equipments by simple MDB modifications without having to re–code the SAS.

The information contained in the physical address field of the GDU is the following:

NODE_ADDRESS	this field contains a string identifying the name of the physical device which is responsible for handling this command
DEVICE_TYPE	this field contains a string identifying the type of device to be used for generating this kind of output to the unit under test, can be used for consistency checking
DEVICE_ADDRESS	this field contains a string identifying the device specific address to be used to 'speak' to the respective device via a dedicated bus, e.g. the device address on a given IEEE488 bus or the RT address on a given MIL–bus
DEVICE_SUBADDRESS	this field contains a string identifying the device specific subaddress to be used to 'speak' to the respective device via a dedicated bus, e.g. the device subaddress on a given IEEE488 bus or the RT subaddress on a given MIL–bus
CHANNEL	this field contains an integer number identifying the channel to which the respective command shall be issued, i.e. typically a channel of an IEEE488 device or an output channel on a dedicated MIL–bus RT.
FUNCTION_CODE	this field contains an integer number identifying the function code to be sent to the specific channel in order to generate the desired output. The function code can be mapped internally to function commands in the SAS, if desired.
PROTOCOL_ID	this field contains an integer number identifying the kind of protocol to be handled between the SAS and the front end device, e.g. "IEEE488 serial poll" or "MIL–Bus mode command with data"
CMD1	this field contains a string identifying the command to be sent to the front end device in order to generate the desired output.
CMD2	this field contains a string identifying another command to be sent to the front end device in order to generate the desired output.

–PREDEFINED_TC / PUS_TC

For the GDUs of type PREDEFINED_TC, the CCSDS packet to be sent to the unit under test.

Using the procedures and functions of the ADT_GDU properly, the stimuli to the unit under test or the front end devices can be generated.

Copyright per DIN 34

Time–tagged GDUs

CGS provides the possibility to time–tag GDU, i.e. they shall be sent at a specific point in time. This feature must, however, be implemented inside the SAS sending the GDU last but not least. As such, part of the GDU sent as a command from CGS to SAS is the time tag specifying at which date and time it has to be sent.

IMPORTANT : see above comments concerning the time tag field in case of CCSDS packets!

It is the responsibility of the SAS to manage GDU queues to ensure that GDUs are sent out in the correct order and at the correct point in time.

Retries of GDUs

Together with the GDU sent out as a command from CGS to the SAS, a re–try parameter is passed to the SAS, indicating how often the SAS shall re–try to send the GDU in case of transmission errors.

It is the responsibility of the SAS to manage GDU re–try queues to ensure that GDUs are sent out in the correct order and with the correct number of re–tries

GDU handling example

In the following an example for handling a CCSDS type GDU is given. This example extracts the CCSDS packet from the GDU and sends it to a front end device.

```
with   TES_DEFINITIONS;          -- the list of withed packages
with   VICOS_DEFINITIONS;
with   ADT_GDU;
with   ADT_GDU_DESCRIPTION;
with   ADT_CCSDS_PACKET;
with   NUMERIC_TYPES;
with   UNCHECKED_CONVERSION;
with   Ada.Text_IO;
...
package GDU_MANAGER                  is          -- specify a package for handling GDUs

      task type T_GDU_MAIN is       -- use a task to handle GDUs
          entry init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID);
          entry send_GDU(GDU : in ADT_GDU.T_GDU);
          entry fini;
      end T_GDU_MAIN;

      for T_GDU_MAIN'storage_size use 400_000;
          -- set the stack size, may be necessary for big GDUs, 400_000 is just a figure

      HANDLER : T_GDU_MAIN;               -- declare a TASK HANDLER

end GDU_MANAGER;                    -- end of the package specification

package body GDU_MANAGER                  is          -- the body of the package
```

Copyright per DIN 34

```
    global_ap_id            : TES_DEFINITIONS.T_APPLICATION_ID;-- some global variables
    global_gdu              : ADT_GDU.T_GDU;
    global_gdu_alternative : ADT_GDU_DESCRIPTION.T_GDU_ALTERNATIVES;
    global_ccsds            : ADT_CCSDS_PACKET.T_CCSDS_PACKET;
    global_send_it          : boolean := false;
    global_out_line         : Ada.Text_IO.file_type;
    global_out_filename     : string(1..11) := "/dev/tty0p6";

    task body T_GDU_MAIN                is
    begin
        accept init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID) do
            global_ap_id := AP_ID;
            Ada.Text_IO.open(global_out_line,Ada.Text_IO.out_file,global_out_filename);
        ...         -- do some more initialisations
        end init;
        loop
            select
                accept send_gdu(GDU : in ADT_GDU.T_GDU) do
                    global_gdu_alternative := adt_gdu.alternative(GDU);          -- which kind of GDU
                    if (global_gdu_alternative = CCSDS_PACKET) then
                        global_ccsds := ADT_GDU.predefined_tc(GDU);          -- extract the CCSDS packet
                        len := ADT_CCSDS_PACKET.user_data_length(global_ccsds);-- the length
                        send_CCSDS_PACKET(global_ccsds,len,out_file);
...                                                                          -- do error checks etc.
                    end if;             -- the CCSDS packet
                end send_gdu;
            or
                accept fini do
                    Ada.Text_IO.close(global_out_line);
                end fini;
            end select;
        end loop;
    end T_GDU_MAIN;

end GDU_MANAGER;
```

F-4 Handling ADUs

ADUs are sent from SAS to CGS. Before that happens, the SAS has to indicate to CGS though that it is able to handle ADUs. For this purpose, the procedure

ANNOUNCE_ADU_SERVICE

from the TES_API has to be called.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–STATUS is the error status returned by this call.

Copyright per DIN 34

After this operation has been called, CGS will eventually send requests for certain ADUs to the SAS, typically as a consequence of the UCL system library command START_ACQUISITION.

As soon as the SAS is no longer in a position to process ADUs, it can call the operation

WITHDRAW_ADU_SERVICE

from the TES_API.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–STATUS is the error status returned by this call.

After this operation has been called, CGS will no longer send new requests for ADUs to the SAS and will inform the user accordingly through appropriate error messages in case certain ADUs would be needed.

A SAS may temporarily change over its local status (ANNOUNCE_ADU_SERVICE, WITHDRAW_ADU_SERVICE, ...) at any time and more than once during a 'session' with CGS.

When a ADU_REQUEST command is then read from CGS, the ADU_DESCRIPTION is obtained from that command using the corresponding function in the ADT_TES_TO_SAS_COMMAND. This ADU_DESCRIPTION can be processed internally in the SAS. At any point in time the SAS is free to send the acknowledge for this command back to CGS.

It is important to note here that it is the responsibility of the SAS programmer/designer to decide, at which point in time the acknowledge for an ADU_REQUEST will be given to CGS. The two extreme cases are:

–The acknowledge is given immediately after the command has been read and it has been decided that reading the command was successful, or

–The command is read, the acquisition of all data related to the ADU_REQUEST can be done because it has been checked that all front end devices are o.k.

Whatever the SAS designer/programmer decides he/she must be aware of the fact that CGS checks for the predefined timeout interval when sending out an ADU_REQUEST to a SAS, i.e. before that interval expires the ACK/NACK must have been received by CGS.

ADU_REQUEST Processing

Similar to the commands sent from CGS to SAS, the ADU_DESCRIPTION being part of the ADU_REQUEST is also private type, encapsulated in an Ada package (The ADU_DESCRIPTION can be extracted from the command using an operation of the ADT_TES_TO_SAS_CMD). This package provides the type definition for the T_ADU_DESCRIPTION plus all operations to read the information needed.

Copyright per DIN 34

The following information is part of the ADU_DESCRIPTION and available to the SAS through proper use of the ADT_ADU_DESCRIPTION:

–DB_SID

The database short identifier (SID) of this ADU_DESCRIPTION. The ADU_DESCRIPTION is the database contents describing the ADU to be sent to CGS.

–PRIVATE_ID

This is an ASCII string defined in the database by the user. Using the private ID, the chicken–egg problem of what has to be defined first, the SAS or the database, can be avoided. The SAS programmer can write the SAS using the private ID to identify which kind of ADU to generate in response to this ADU_REQUEST. Also, the database can be filled with ADU_DESCRIPTION at any time using a predefined set of private IDs. At no point in time must an SAS programmer know the SID of a dedicated ADU_DESCRIPTION. This is extremely important, if the SAS programmer decided not to interpret the ADU_REQUEST with respect to the individual measurements contained in it but instead the SAS generates a predefined datapacket.

–ALTERNATIVE

There are four ADU alternatives defined currently in CGS:

STRUCTURED	a structured ADU already contains a list of CGS compatible raw values in the CGS raw value data format. This kind of ADU is the preferred one if the SAS reads individual values of individual enditems from individual front ends.
UNSTRUCTURED	in this case the SAS reads a binary data buffer from a front end and simply puts the whole buffer into an ADU. Subsequently, CGS will unpack the data from the binary buffer, do the raw value extraction and convert the bits and pieces into CGS compatible raw values for further processing.
CCSDS_PACKET	a CCSDS telemetry packet available as a binary data packet in the SAS. Subsequently, CGS will unpack the data from the CCSDS packet, do the raw value extraction and convert the bits and pieces into CGS compatible raw values for further processing.
PUS_PACKET	a PUS telemetry packet available as a binary data packet in the SAS. Subsequently, CGS will unpack the data from the CCSDS packet, do the raw value extraction and convert the bits and pieces into CGS compatible raw values for further processing.
UNDEFINED	an undefined type (not used)

–ACQUISITION_RATE

The acquisition rate in terms of milliseconds for the ADU to be provided to CGS. 1000 indicates the ADU shall be sent every second. A value of 0 indicates that the ADU shall be sent whenever possible, i.e. typically this is the value for a CCSDS packet since it is unknown when a specific packet will be received on ground. CGS itself does not check the acquisition frequency of an SAS.

–GLOBAL_ADDRESS

The low level device specific information needed by the SAS to really acquire the data for the corresponding ADU from the font end devices. This information is stored in the configuration database and passed to

Copyright per DIN 34

the SAS by CGS. CGS does not interpret it. It is up to the SAS to decide, how the information in the physical address field should be used. It is, however, good practice to re–use as much from the information contained herein, in order to ensure consistency enforced by the use of the configuration database also in the SAS area and in order to simplify reconfiguration of front end equipments by simple MDB modifications without having to re–code the SAS.

The information contained in the physical address field of the ADU_DESCRIPTION is the following:

NODE_ADDRESS	this field contains a string identifying the name of the physical device which is responsible for handling this command
DEVICE_TYPE	this field contains a string identifying the type of device to be used for generating this kind of input from the unit under test, can be used for consistency checking
DEVICE_ADDRESS	this field contains a string identifying the device specific address to be used to 'speak' to the respective device via a dedicated bus, e.g. the device address on a given IEEE488 bus or the RT address on a given MIL–bus
DEVICE_SUBADDRESS	this field contains a string identifying the device specific subaddress to be used to 'speak' to the respective device via a dedicated bus, e.g. the device subaddress on a given IEEE488 bus or the RT subaddress on a given MIL–bus
CHANNEL	this field contains an integer number identifying the channel from which the respective data shall be read, i.e. typically a channel of an IEEE488 device or an input channel on a dedicated MIL–bus RT.
FUNCTION_CODE	this field contains an integer number identifying the function code to be sent to the specific channel in order to generate the desired input. The function code can be mapped internally to function commands in the SAS, if desired.
PROTOCOL_ID	this field contains an integer number identifying the kind of protocol to be handled between the SAS and the front end device, e.g. "IEEE488 serial poll" or "MIL–Bus mode command with data"
CMD1	this field contains a string identifying the command to be sent to the front end device in order to generate the desired input.
CMD2	this field contains a string identifying another command to be sent to the front end device in order to generate the desired input.

The GLOBAL_ADDRESS information must not be present in all ADU_DESCRIPTIONs. It only makes sense, if all data belonging to the related ADU can be acquired from one device, one subaddress and one channel. This is the case for the unstructured ADUs and the CCSDS–packet ADUs since the data buffer (the CCSDS packet) typically comes from exactly one location. In the other case, each individual enditem has its private physical address information (see below)

Copyright per DIN 34

–ENDITEM_DESCRIPTION

This information identifies which measurement enditems defined in the mission database have to be put into this ADU. The following type of information is comprised in the ADU_DESCRIPTION:

SID	the short identifier of the respective measurement
RAW_VALUE_TYPE	the identification of the CGS provided raw value type, which in fact is an enumeration type with the following alternatives: UNDEFINED SINGLE_BIT GROUP_OF_BITS INTEGER FLOAT BYTE_STREAM This information is only available for structured ADUs since only then the SAS has to put in raw values
INDEX	The index position of this raw value in the list of raw values of a structured ADU, e.g. 10 indicates this raw value is the tenth in the list.
PARAMETER_ADDRESS	the physical address information specifying how to obtain the data. The contents of this field is the same as for the GLOBAL_ADDRESS field of the ADU_DESCRIPTION.

–CCSDS_ID

This field indicates the CCSDS application ID of the CCSDS packet to be encapsulated in this ADU.

Providing ADUs to CGS

The processing of an ADU includes the typical processing steps, after having read the ADU_REQUEST command and extracted the ADU_DESCRIPTION from it:

–determine the type of ADU to be provided by checking the PRIVATE_ID of the ADU_DESCRIPTION

–establish an Ada variable holding the ADU by using the CONSTRUCT operation from ADT_ADU with the ADU_DESCRIPTION as an input

–start the acquisition of data from the front end equipment(s) for those data to be put into the ADU

–fill the data into the ADU by using the appropriate operation from ADT_ADU:

SET_RAW_VALUE:	adds an individual raw value to the ADU. Individual raw values can be added until the list of required values has been completed
SET_BINARY_BUFFER:	puts the binary buffer into the ADU
SET_CCSDS_PACKET:	puts the CCSDS packet into the ADU

–set the time tag of the ADU using the operation SET_TIME_TAG of the ADT_ADU. The time tag should carry the actual time (local + SMT) when the raw data have been acquired. For CCSDS packets the time tag from the packet should be extracted.

–set the ADU sequence counter using the operation SET_SEQUENCE_COUNT from ADT_ADU. All ADUs have a sequence counter which has to be set and maintained by the SAS.

Copyright per DIN 34

–set the data interruption flag of the ADU using the operation SET_DATA_INTERRUPTION from ADT_ADU. All ADUs have a data interruption indication which is normally set to false, i.e. no data has been lost between two subsequent ADUs. Should the SAS detect, however, that between two subsequent ADUs there has been a data interruption (for whatever reason, e.g. a CCSDS packet has been lost, a device has not responded in time, some raw values were bad,...) the first ADU after the data interruption should be flagged with DATA_INTERRUPTION set to TRUE.

–send the ADU to CGS at the predefined time interval using the procedure

SUPPLY_ADU

from the TES_API package.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ADU is the ADU to be supplied

–STATUS is the error status returned by this call.

Suspension of ADU delivery

In the case where the application is no longer able to provide a given ADU to CGS, it can indicate that the ADU delivery is suspended. This will lead to have all mesaurements from the ADU to become the status *STATIC*, until the delivery resumes. A specific operation allows to indicate the suspension of the delivery, while it is resumed simply by calling the next time SUPPLY_ADU for that ADU.

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ADU is the suspended ADU

–STATUS is the error status returned by this call.

ADU handling example

In the following example, an ADU handling example is shown. The SAS waits for an ADU request and then sends a predefined ADU of type CCSDS packet without parsing the ADU description contained in the ADU request. The input data from the front end is assumed to be a binary block.

```
with   TES_DEFINITIONS;          -- the list of withed packages
with   VICOS_DEFINITIONS;
with   CGS_CALENDAR;
with   ADT_GDU;
with   ADT_ADU;
with   ADT_ADU_DESCRIPTION;
with   ADT_CCSDS_PACKET;
```

Copyright per DIN 34

```
with   CALENDAR;
with   NUMERIC_TYPES;
with   TES_API;
with   UNCHECKED_CONVERSION;
with   Ada.Text_IO;


package ADU_MANAGER                is        -- define a package for ADU handling

       task type T_ADU_MAIN is      -- define a task type for ADU handling
           entry init (AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID);
           entry insert_adu (ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION);
           entry delete_adu (ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION);
           entry fini;
       end T_ADU_MAIN;

       for T_ADU_MAIN'storage_size use 400_000; -- set the stack size, 400_000 is just an example

       HANDLER : T_ADU_MAIN;                    -- define a task HANDLER

end ADU_MANAGER;                                -- end of the package spec.

package body ADU_MANAGER                is

       max_dde_i_length : integer := 255;
       packet_size      : integer := 1024;

       subtype    T_UDATA_BUFFER      is NUMERIC_TYPES.byte_array(1..packet_size);
       subtype    T_DDE_INPUT_BUFFER  is string(1..max_dde_i_length);
       subtype    T_BA4               is NUMERIC_TYPES.BYTE_ARRAY(1..4);

       timetag                  : CGS_CALENDAR.T_DATE_AND_TIME;
       global_ap_id             : TES_DEFINITIONS.T_APPLICATION_ID;
       global_adu_description   : ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION;
       global_send_it           : boolean := false;
       global_ccsds             : ADT_CCSDS_PACKET.T_CCSDS_PACKET;
       udata                    : T_UDATA_BUFFER := (others => 0);
       global_adu               : ADT_ADU.T_ADU;
       adu_seq_nb               : integer := 1;
       rtn                      : TES_DEFINITIONS.T_RETURN_STATUS;
       global_delay             : duration;
       dde_data                 : T_DDE_INPUT_BUFFER;
       act_dde_len, acq_rate_in_msec : natural;

       task body T_ADU_MAIN            is
           begin
               accept init(AP_ID: in TES_DEFINITIONS.T_APPLICATION_ID) do
                   global_ap_id := AP_ID;
                       -- set up the CCSDS packet here
                   ADT_CCSDS_PACKET.set_packet_type(global_ccsds,ADT_CCSDS_PACKET.tm);
                   ADT_CCSDS_PACKET.set_application_id(global_ccsds,41);
                   ADT_CCSDS_PACKET.set_sequence_count(global_ccsds,1);
                   ADT_CCSDS_PACKET.set_length(global_ccsds,packet_size);
                   . . .
                   ADT_CCSDS_PACKET.set_packet_checksum_type
                       (global_ccsds,
                        ADT_CCSDS_PACKET.no_checksum);
                   timetag := CGS_CALENDAR.get_date_and_time(CALENDAR.CLOCK);
```

Copyright per DIN 34

```
            ADT_CCSDS_PACKET.set_time_tag(global_ccsds,timetag);
            ADT_CCSDS_PACKET.set_user_data(global_ccsds,udata);
        end init;
        loop
            select
                accept insert_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION) do
                    acq_rate_in_msec := ADT_ADU_DESCRIPTION.ACQUISITION_RATE(ADU);
                    global_send_it := true;
                    ADT_ADU.construct(global_adu,ADU); -- create the ADU
                end insert_adu;
            or
                accept   delete_adu(ADU : in ADT_ADU_DESCRIPTION.T_ADU_DESCRIPTION) do
                    Ada.Text_IO.PUT_LINE("Removing an ADU");
                    global_send_it := false;
                end delete_adu;
            else
                delay(global_delay);
                -- read data from the front end device here
                if global_send_it then
                    interprete_input_data(udata); -- udata block returned
                    ADT_CCSDS_PACKET.set_user_data(global_ccsds,udata); -- put block into  CCSDS
                    timetag := CGS_CALENDAR.get_date_and_time(CALENDAR.CLOCK);
                    ADT_CCSDS_PACKET.set_time_tag(global_ccsds,timetag);
                    ADT_ADU.set_time_tag(global_adu,timetag);
                    ADT_ADU.set_sequence_number(global_adu,adu_seq_nb);
                    adu_seq_nb := adu_seq_nb + 1;
                    ADT_ADU.set_ccsds_packet(global_adu,global_ccsds);
                    TES_API.supply_adu(global_ap_id,global_adu,rtn);
                    if integer(rtn) /= integer(TES_DEFINITIONS.SUCCESS) then
                        Ada.Text_IO.PUT_LINE
                            ("After sending ADU, RTN="
                                & TES_DEFINITIONS.T_RETURN_STATUS'image(rtn));
                    end if;
                end if;
            end select;
        end loop;
    end T_ADU_MAIN;
end ADU_MANAGER;
```

F-5 Reading End Item Data from CGS

SASes have the possibility of reading individual enditem data from CGS. Raw data as well as engineering data can be provided. This mechanisms simplifies the access to individual enditem data in case the SAS only seldom needs rare enditem data.

CGS delivers raw and engineering data to SASes out of its internal data pool, i.e. the most recent value is delivered. However, the delivery of individual raw data enditem values to SAS is not synchronised with the delivery of ADUs by SAS. A raw value is always delivered immediately out of the data pool to the SAS if it is currently available, that is it has been acquired before and the acquisition has not been stopped in the mean time. The similar principle applies also for delivering engineering data

The procedure to read a raw data value from CGS is the

READ_RAW_VALUE

Copyright per DIN 34

operation from the TES_API.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ENDITEM_PATHNAME is the enditem pathname of the raw value. The enditem has to refer to an EGSE_xxx_MEASUREMENT in the database since only the measurements have raw values.

–VALUE_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

–INT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

–UINT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is unsigned integer, in other cases this value is undefined;

–FLOAT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

–DFLOAT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is double float, in other cases this value is undefined;

–BYTESTREAM_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

–STATUS is the error status returned by this call.

The procedure to read a measurement value from CGS is the

READ_MEASUREMENT

operation from the TES_API.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ENDITEM_PATHNAME is the enditem pathname of the value. A reference to an enditem of type EGSE_xxx_MEASUREMENT is allowed. The enditem is identified through its database pathname. For simplicity reasons this is a string

–SID is the short identifier (SID) of the value. A reference to an enditem of type EGSE_xxx_MEASUREMENT is allowed. The enditem is identified through its SID.

–VALUE_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

–INT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

Copyright per DIN 34

–UINT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is unsigned integer, in other cases this value is undefined;

–STATECODE_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is STATE_CODE, in other cases this value is undefined;

–FLOAT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

–DFLOAT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is double float, in other cases this value is undefined;

–BYTESTREAM_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

–RAW_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

–RAW_INT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

–RAW_UINT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is unsigned integer, in other cases this value is undefined;

–RAW_FLOAT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

–RAW_DFLOAT_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is double float, in other cases this value is undefined;

–RAW_BYTESTREAM_VALUE is the raw value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

–STATUS is the error status returned by this call.

A SAS may read any raw or engineering value available in a distributed CGS test system configuration. CGS will internally manage to obtain the value from the place where it is maintained.

The procedure to read an engineering value from CGS is the

READ_ENGINEERING_VALUE

operation from the TES_API.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ENDITEM_PATHNAME is the enditem pathname of the value. A reference to an enditem of type EGSE_xxx_MEASUREMENT or EGSE_xxx_SW_VARIABLE is allowed. The enditem is identified through its database pathname. For simplicity reasons this is a string

Copyright per DIN 34

–VALUE_ALTERNATIVE indicates the type of the enditem value to be read, e.g. whether it is an INTEGER or a FLOAT.

–INT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is integer, in other cases this value is undefined;

–UINT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is unsigned integer, in other cases this value is undefined;

–STATECODE_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is STATE_CODE, in other cases this value is undefined;

–FLOAT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is float, in other cases this value is undefined;

–DFLOAT_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is double float, in other cases this value is undefined;

–BYTESTREAM_VALUE is the engineering value returned by CGS in case the VALUE_ALTERNATIVE is byte stream, in other cases this value is undefined;

–STATUS is the error status returned by this call.

A SAS may read any raw or engineering value available in a distributed CGS test system configuration. CGS will internally manage to obtain the value from the place where it is maintained.

F-6 Providing End Item Data to CGS

SASes can write into SW variables maintained by CGS. SW variable have to be defined in the configuration database. For this purpose, the operation

WRITE_ENGINEERING_VALUE

from the TES_API has to be used.

The parameters of these operations have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ENDITEM_PATHNAME is the enditem pathname of the value. A reference to an enditem of type EGSE_xxx_SW_VARIABLE is allowed.

–VALUE is the engineering value to be written CGS. The overloaded versions of the procedure WRITE_ENDITEM_VALUE have different types for the formal parameter VALUE depending on the type of value to be written into the enditem. Care has to be taken that the correct type of enditem data is written into an enditem, otherwise exceptions will occur

–STATUS is the error status returned by this call.

Copyright per DIN 34

When a SAS writes to a SW variable, the monitoring of that variable will be done by CGS afterwards (if defined so in the configuration database).

F-7 Starting Parameterless UCL Automated Procedure

Allow to start an UCL automated procedure without parameters on connected testnode.

START_AP

The parameters of this operation have the following meaning:

–The Application_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–AP_NAME is the enditem pathname of the UCL automated procedure. The enditem has to refer to an enbditem in the database.

–STATUS is the error status returned by this call.

F-8 Exchanging Messages with APs

SAS can read messages from APs and can also send messages to them. Messages are simple ASCII strings.

When an AP wants to send a message to a SAS it uses a UCL system library procedure. The result of calling this procedure is that a command is sent to the SAS, containing the message as a string. Of course, the SAS must be running at the point in time the message is sent to it since they are not buffered.

At any point in time the user can also call the corresponding UCL system library procedure from the HLCL command window and thus send a message to a SAS.

The procedure for sending a message from a SAS to an automated procedure is

SEND_MESSAGE_TO_AP

from the TES_API.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–AP_NAME is the pathname of the automated procedure to which the message shall be sent. This AP must be running on the local test node which this SAS logically ”belongs” to. If more than one instance of the AP is running, the message will be passed to all APs.

–MESSAGE is the message to be written to the AP. The type definition in TES_DEFINITIONS enforces a length of 80 characters for the message (left justified, filled with blanks if message is shorter).

–MESSAGE_LENGTH is the actual length of the message.

–STATUS is the error status returned by this call.

Copyright per DIN 34

It is good programming practice that an AP should pass its name to the SAS it wants to exchange messages with and then the SAS should only use those AP names previously registered.

F-9 Reading the CGS Time

The CGS concept foresees that all computer clocks of all machines involved in a test are synchronised by the Time Synchronisation Software (TSS) to an accuracy of 1..2 msec. The computer clocks run the local time (wall clock). A SAS could therefore use the standard Ada mechanisms to obtain the local time. Also, delays based on local time can be achieved in this way. In parallel to the local time, CGS maintains a simulated mission time (SMT) which is under user control. At any point in time the SMT can be stopped, started or reset. Once it is running, it runs with a constant offset in parallel to the local time.

TES_API provides one operation to read the local time plus the simulated mission time SMT. This operation is called

READ_TIME

in the TES_API.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–LOCAL_TIME is the local time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds.

–SMT is the simulated mission time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds.The SMT may return a special value (CGS_CALENDAR.NULL_DATE_AND_TIME) in case it is not initialised.

–SMT_RUNNING indicates, if the SMT is counting or stopped.

–STATUS is the error status returned by this call.

TES_API provides another operation to read the details of the SMT. This operation is called

READ_SMT_DETAILS

in the TES_API.

This procedure returns the detailed status of the SMT.

– The application id is the one returned by TES during the connect operation.

–SYNC_SW_STATUS returns, whether the NTP Status file, which is updated every, minute by a crontab job if NTP is running, was read and has the correct format

– SYNC_SW_LOCKED returns, whether synchronisation source for the local NTP process is considered as reliable.

Copyright per DIN 34

–SYNC_SW_IN_LIMIT returns, whether the time deviation between the NTP client and its synchronisation source is in the allowed limit

–SMT_SERVER returns, whether the local machine is SMT server (TRUE) or not

–MTU_PRESENT returns, whether a MTU is present (TRUE) or not

–SMT_STATUS returns the actual SMT status

–SMT_OFFSET returns the actual SMT offset

–MTU_STATUS returns the actual MTU status

–STATUS is the error status returned by this call.

Should the SAS want to delay until a given local time or SMT, the operation

WAIT_UNTIL

from the TES_API has to be used.

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–WAKE_UP_TIME is the time in the CGS_CALENDAR layout, i.e. three integer numbers specifying the year, month and day plus four integer numbers indicating the hours, minutes, seconds and milliseconds at which the SAS shall be activated again.

–TIME_TYPE is an enumeration type allowing the two alternatives LOCAL_TIME and SMT only.

–STATUS is the error status returned by this call.

This procedure will block until the desired wake–up time has expired and then return successfully (normally). Should an SMT wait–until be desired and should the SMT be reset in the meantime, the procedure will return immediately after the reset of the SMT.

F-10 Reporting Errors and Passing Messages to CGS

At any point in time the SAS might detect errors and / or might want to pass a message to CGS. For this purpose, the procedure

SEND_ERROR_MESSAGE

in the TES_API has to be used.
Messages sent from a SAS to CGS will be displayed in the CGS error window and they will be logged in the test result database.
NOTE: For historic reasons the procedure is called SEND_ERROR_MESSAGE although a SAS can also send normal information messages..

The parameters of this operation have the following meaning:

Copyright per DIN 34

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–ERROR_MESSAGE_SHORT is a 40 character string containing a comprehensive message text. The CGS message window can be configured to always show the short text.

–ERROR_MESSAGE_LONG is a 255 character string containing a full message text. The CGS message window can be configured to always show the long text or suppress it and show it on demand only..

–ERROR_TYPE is an indication for the type of message to be sent. It is a four character string that can be used under user control. It is however recommended to use only the following values as predefined in package VICOS_DEFINITIONS:

```
UUT_LOG_GROUP           : constant string := "UUT "; -- msg received from UUT
INFO_MESSAGE_TYPE       : constant string := "INFO";
GENERAL_MESSAGE_TYPE    : constant string := "MSG ";
ERROR_MESSAGE_TYPE      : constant string := "ERR ";
WARNING_MESSAGE_TYPE    : constant string := "WRN ";
EXCEPTION_MESSAGE_TYPE  : constant string := "EXC ";
ALARM_MESSAGE_TYPE      : constant string := "ALRM";
```

The value "SAS " would also be allowed. In the test result database, this ERROR_TYPE is later on used for selecting certain messages.

–LOCAL_TIME_TAG is the CGS_CALENDAR like time tag for this message in terms of local time (wall clock time). The SAS must not provide the time–tag. In this case the API will add the current time as the time tag.

–SMT_TIME_TAG is the CGS_CALENDAR like time tag for this message in terms of SMT. The SAS must not provide the time–tag. In this case the API will add the current time as the time tag.

–SHOW_IT is a boolean flag which indicates whether or not TES will route the error message to the user interface. In any case, the message will be logged.

–STATUS is the error status returned by this call.

F-11 Store / Retrieve Files to Test Resul Data Base (TRDB)

Allow to store files or retrieve filenames from / to test result database (TRDB).

STORE_USER_FILE

The parameters of this operation have the following meaning:

–The FILE_NAME is the complete unix pathname to location of the file to archive.

–FILE_TYPE is the the type of user file.

–SESSION_NAME is the name of the currently active session to which the file has been assigned.

–STATUS is the error status returned by this call.

RETRIEVE_USER_FILE

The parameters of this operation have the following meaning:

Copyright per DIN 34

–The SESSION_NAME is the name of the archive session, where the file is stored.

–The FILE_NAME is the basename of the user file name of a previously stored file.
Note: The file name must only contain the basename of the file.

–LOCATION is the complete pathname of the user file within the TRDB. Note: the file will **not** be copied, only the pathname will be returned.

–STATUS is the error status returned by this call.

F-12 Get GDU Description from CGS

Allow to read a GDU description from CGS.

GET_GDU

The parameters of this operation have the following meaning:

–The ENDITEM_PATHNAME is the pathname of the GDU.

–The SID is the short identifier of the GDU.

–In case of invalid ENDITEM_PATHNAME or invalid SID (not exist or not of correct type) the exception GDU_TYPE_MISMATCH will be raised.

–The returned value is the GDU.

F-13 Translation SID – Pathname/Nickname

Allow to translate pathnames to SID to nickname.

PATHNAME_TO_SID

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–The ENDITEM_PATHNAME is the pathname of the enditem.

–The returned value is the SID.

–In case of any error a NULL_SID is returned.

NICKNAME_TO_SID

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–The NICKNAME is the nickname of the enditem.

–The returned value is the SID.

–In case of any error a NULL_SID is returned.

Copyright per DIN 34

GET_PATHNAME

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–The SID is the short identifier of the enditem.

–The returned value is the string of the pathname.

–In case of any error an empty string is returned.

GET_NICKNAME (from ENDITEM_PATHNAME)

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–The ENDITEM_PATHNAME is the pathname of the enditem.

–The returned value is the string of the nickname.

–In case of any error an empty string is returned.

GET_NICKNAME (from SID)

The parameters of this operation have the following meaning:

–The APPLICATION_ID is the one returned from the respective CONNECT at the begin of the session with CGS.

–The SID is the short identifier of the enditem.

–The returned value is the string of the nickname.

–In case of any error an empty string is returned.

Copyright per DIN 34

G INDEX OF ALL USER MANUAL PROCEDURES

” Creating a command tool window 4–4
” Saving your desktop workspace 4–4
” Setup the Message Handler application 4–24
” View the Message Info 4–31
” Starting a Mission Configuration Session 5–4
” Using Scroll-Bars in I_MDB Navigator 5–7
” Navigating down within a System Tree 5–7
” Navigating down within a User Tree Version 5–8
” Navigating down to a CCU version from the complete list 5–9
” Navigating to a CCU of a particular System Tree Node 5–9
” Navigating upwards in the Element Configuration Tree 5–10
” Checking CM Status 5–11
” Creating initial Configuration Data Unit (CDU) 5–11
” Creating a new CDU Version 5–12
” Creating User Trees Nodes 5–14
” Create a new Configuration Control Unit (CCU) 5–14
” Create a CCU Version 5–15
” Specifying CCU Version Contents 5–17
” Creating an End Item (exemplary of type General Purpose) 5–19
” Accessing Operations on End Items. 5–21
” Creating an Automated Procedure 5–24
” Accessing Automated Procedures 5–25
” Compiling an Automated Procedure 5–26
” Finding Error Locations 5–27
” Generation of a Compiler Listing 5–27
” Make Mode 5–28
” Generation of an Error List File 5–29
” Choosing the Compiler’s Optimization Mode 5–29
” Storing an Automated Procedure 5–29
” Viewing the Action Log 5–30
” Getting Information about an AP 5–30
” Getting Syntax Help 5–32
” Getting Pathname Help 5–33
” Create HLCL Command Sequence in Unix file system 5–37
” Edit HLCL Command Sequence in Unix File System 5–37
” Defining a DB Server Node 6–6

Copyright per DIN 34

" Defining the Test Configuration – Database and Simulator Node 6–7
" Defining the Test Configuration – Workstation Nodes 6–9
" Defining the Test Configuration – Test Nodes 6–10
" Defining the Test Configuration – Test Node Items 6–13
" Defining the Test Configuration – Test Node SASs 6–14
" Synchronous SAS Main Program Structure 6–35
" Pseudo asynchronous SAS Main Program Structure 6–36
" Asynchronous TES SAS Main Program Structure 6–37
" Preparing the SAS environment and Implementing DBS SAS (on SUN) 6–40
" Example of Expression for an Integer_Derived_Value: 6–65
" Setting a Default Path 7–78
" Invoking the Flags Window 7–78
" Selecting Explicitely a Function from a Pop–Up Menu 8–6
" Activating the Default Function from a Pop–Up Menu 8–6
" List Existing Evaluation Sessions 8–7
" Select an Evaluation Session 8–8
" Create an Evaluation Session 8–9
" Select Execution Sessions 8–12
" Select a CCU from a Selected Session 8–15
" Select the CCU from the MDB 8–16
" Select the CCU from the List of CCUs Used by the Default Test Session 8–17
" Validate the Selection of Execution Sessions and CCU 8–17
" Archive a Session 8–20
" Retrieve a Session 8–21
" Delete On–Line Data 8–22
" Export an Evaluation Session 8–25
" Export an Execution Session 8–26
" Import a Session 8–26
" Select a Predefined Definition 8–29
" Save a Definition 8–29
" Exec & Display a Definition 8–30
" Enter directly a Time Frame. 8–31
" Select a Time Frame from an Execution Session 8–31
" Select a Time Frame from two User Events 8–32
" Use of a Specific Normal Criterion 8–35
" Specify an Events Logging Definition 8–36
" Generate an ADT Result 8–38
" Create a Definition 8–40
" Create a Definition (continue) 8–41

Copyright per DIN 34

” Consult the Packets Dumped in the Archive Files 8–42
” Select the Output Format for the Packets 8–44
” Reformat a Raw Data Packet 8–44
” Consult the Packets dumped in the Archive Files in a Summary Format 8–45
” Save a Dump in a Result File 8–45
” Select the Every n Sampling Mode 8–48
” Select the Time based Sampling Mode 8–48
” Define a Parameters Set 8–49
” Generate the Data Set 8–51
” Generate a listing 8–52
” Build a Statistics Generation Definition 8–55
” Build a Graph Display Definition 8–60
” Edit the Line Graph Properties 8–61
” Edit the XY Graph Properties (see Figure 8–41) 8–63
” Create a Graphic 8–65
” Store a File in the Test Result Database (i.e in a Test Evaluation Session) 8–68
” Delete one or more Files from the Working Directory 8–69
” Copy a Single File 8–69
” Copy Several Files At Once 8–70
” priv 10–8
” priv mdb pathname 10–8

Copyright per DIN 34

H TEST AND CONTROL RELATED HLCL COMMANDS

H-1 Basics

The High Level Command Language (HLCL) is the interactive counterpart of the User Control Language. (UCL). UCL acts as a pure programming language: automated procedures and libraries are edited off–line, compiled and kept in the database for later on–line execution.

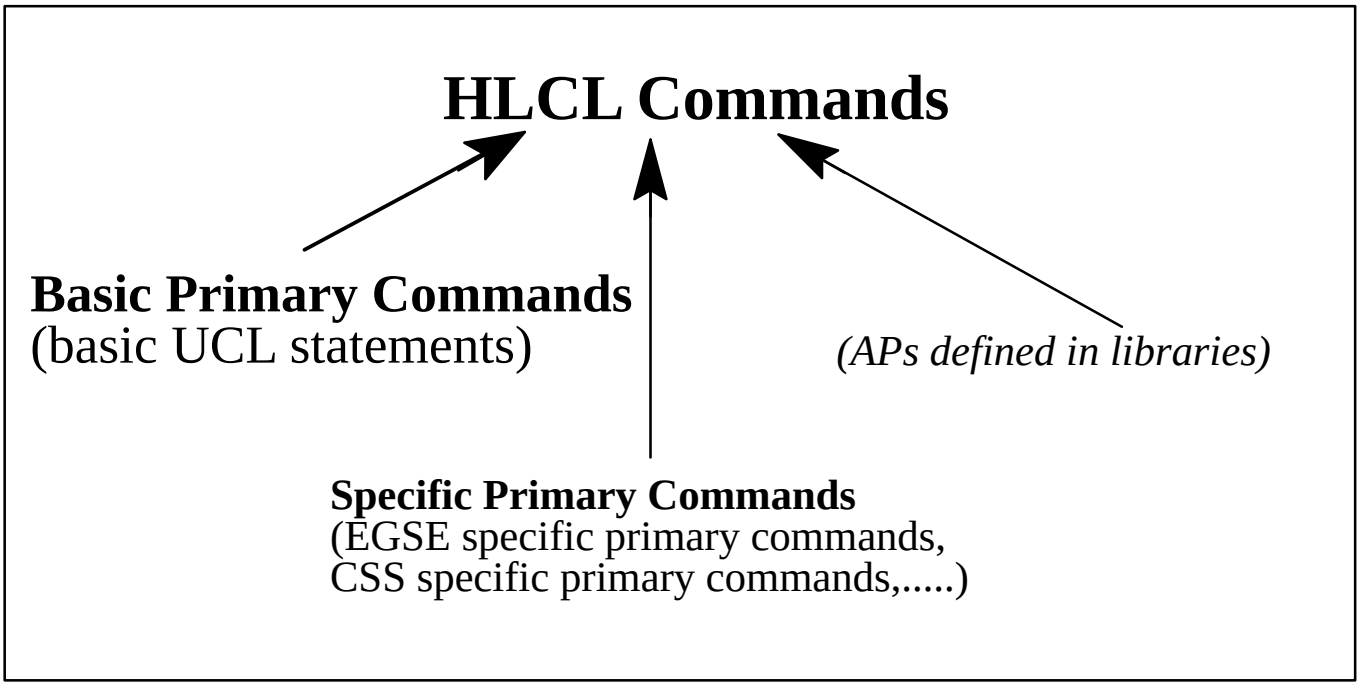


Figure 20 : *Different Types of HLCL Commands*

HLCL provides most of the basic language material of UCL, as well as HLCL specific extensions.

- The basic UCL statements constitute the *basic primary command* set of HLCL.
- Additionally there are the *specific primary commands*. These are commands specific to certain products, i.e. commands which will be followed by CSS only. These commands are configuration dependent, i.e. in a configuration without CSS the CSS related commands are not available.
- HLCL has the capability of dynamically extending its primary command and function set by inheriting new commands and functions from the Columbus database.

H-2 HLCL Language Definition and Primary Commands

The HLCL language definition and the primary commands are described in the HLCL Reference Manual. See chapter 2 for detailed reference.

Copyright per DIN 34

H-3 Application Specific Primary Commands

H-3.1 EGSE Specific Primary Commands

The following list gives the list of statements that are specific user operation related commands required to control VICOS S/W

ABR_AP

Function: Abort an automated procedure

Parameters:

ap	Mode:	Mandatory
	Type:	Integer
	Meaning:	process number of the AP process
node	Mode:	Optional
	Type:	Pathname (EGSE_NODE)
	Default:	\\
	Meaning:	path name of the test node where the AP is running

Example: `ABORT 3, \NODES\TN_1`

ABR_ALL_APS

Function: Abort all automated procedure running on the addressed testnode

Parameters:

node	Mode:	Optional
	Type:	Pathname (EGSE_NODE)
	Default:	\\
	Meaning:	path name of the test node where the APs are running

Example: `ABORT_ALL_APS`

ASIGN_PICTURE

Function: Create a synoptic display and assign the picture to it

Parameters:

picture	Mode:	Mandatory
	Type:	Pathname
	Meaning:	Name of the picture to be displayed
width	Mode:	Optional
	Type:	Integer
	Default:	–1
	Meaning:	Width of the synoptic window to be created
height	Mode:	Optional
	Type	Integer
	Default:	–1
	Meaning:	Height of the synoptic window to be created
horizontal_position	Mode:	Optional
	Type	Integer

Copyright per DIN 34

Default: –1
Meaning: X position of the synoptic window to be created

vertical position
Mode: Optional
Type Integer
Default: –1
Meaning: Y position of the synoptic window to be created

Example:
```
ASSIGN_PICTURE \DMS_05\PICTURE07
ASS_P          \DMS_05\PICTURE07,400,400,100,0
```

GET_ENVIRONMENT

Function: Obtain value for environment name

Parameters:

name
Mode: Mandatory
Type: String
Meaning: Name of the environment variable.

value
Mode: Mandatory
Type: String
Meaning: Value of the environment variable.

Example:
```
VARIABLE SAS_HOME: STRING (100);
GET_ENVIRONMENT ”SAS_HOME”, SAS_HOME
```

HISTØR

Function: Display a list of previously entered commands

Parameters: count
Mode: Optional
Type: Integer
Default: 10
Meaning: Number of commands to be displayed

Constraints: in command facility only

Example:
```
HISTORY 25
```

Copyright per DIN 34

LOAD_UCL

Function: Load UCL item on an EGSE node.

Parameters:			
	ITEM	Mode:	Mandatory
		Type:	UCL_ITEM_NAME
	NODE	Mode:	Optional
		Type:	EGSE_NODE
		Default:	\\
		Meaning:	Node on which the item will be loaded.
	STATUS	Mode:	Optional
		Type:	INTEGER
		Meaning:	Return status (UCL_RETURN, 1 is success).

*) This command is in HLCL command window and in synoptic display available, not in HCI tree explorer.

PRINT

Function: Print a file

Parameters:			
	file	Mode:	Mandatory
		Type:	String
		Meaning:	File to be printed (including directory path)
	printer	Mode:	Optional
		Type:	enumeration type
		Default:	LASER_PRINTER1
		Meaning:	Name of the printer which prints the document

Example: `PRINT ”vicos/testfile”`

Copyright per DIN 34

START_HCI_APPLICATION

Function: Start an application program or HCI window application on the workstation.

Parameters:

application	Mode:	Mandatory
	Type:	String
	Meaning:	Name of the program. If the application name has a **HCI.** prefix the HCI window application is started (see example 2). The list of HCI window applications is described in 7.3.2.5 Screen Setup Maintenance.
parameter	Mode:	Optional
	Type:	String
	Default:	""
	Meaning:	If the application needs parameters, they can be given here.
user_confirmed		
	Mode:	Optional
	Type:	Boolean
	Default:	True
	Meaning:	If true, the user is ask for confirmation before starting the application.

Example 1:
```
START_HCI_APPLICATION    "${OPENWINHOME}/bin/xterm","",false
```

Example 2:
```
START_HCI_APPLICATION    "HCI.GRAPH_FACILITY    -MEASUREMENT
\MOTOR\CURRENT -GRAPH 1 -MINIMUM 0.0 -MAXIMUM 100.0", "", false
```

Example 3:
```
start_hci_application                                    APPLICATION:
"$CGSI_HOME/bin/linuxi/acknowledge",            APPLICATION_PARAMETER:
"""SOL_SA9_Counter is out of sequence!"" ""Please check values in
synoptic.""", USER_CONFIRMED: false;
```

START_UCL_DEBUGGER

Function: Start UCL Debugger.

Parameters:

ITEM	Mode:	Mandatory
	Type:	UCL_AUTOMATED_PROCEDURE
	Meaning:	UCL automated procedure loaded into debugger.
NODE	Mode:	Optional
	Type:	EGSE_NODE
	Default:	\\
	Meaning:	Node the debugger will connect to.
STATUS	Mode:	Optional
	Type:	INTEGER
	Meaning:	Return status (POSIX error codes, 0 is success).

*) This command is in HLCL command window and in synoptic display available, not in HCI tree explorer.

Copyright per DIN 34

H-3.2 CSS Related Specific Primary Commands

The following commands are to be handled by CSS. Within a combined CSS/VICOS simulator configuration the commands may be entered either within the CSS environment (through MOCS if CSS is running in stand–alone mode) or within the VICOS environment (through HCI).

```
------------------------------------------------------------------------
--
-- ***************************
--  System UCL System Library
-- ***************************
--
-- ABSTRACT
--     Defines CSS specific HLCL commands.
--     This library is for use in HLCL procedures only.
--
--     Must be compiled with Body Id = 0
--                       for Language = HLCL.
--
--     recommended Nickname: "HLCL_CSS_CMDS"
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : System System-Library
--                              CGS     CM                                      :     "$Id:
//cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/hlcl_css_cmds_.ucl#5 $"
--*---------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library HLCL_CSS_CMDS;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type ACTIVATION = (STEPWISE, CONTINUOUS);

   type CSS_FB_DATABASE_ITEMS = pathname (TOPLEVEL_COMPOSITE_FB,
                                          COMPOSITE_FB,
                                          CONSTANT_FB,
                                          ASYNCHRONOUS_FB,
                                          SYNCHRONOUS_FB);

   type CSS_FB_DATABASE_ITEMS_WITH_DEFAULT = pathname (TOPLEVEL_COMPOSITE_FB,
                                                       COMPOSITE_FB,
```

Copyright per DIN 34

```
                                                    CONSTANT_FB,
                                                    ASYNCHRONOUS_FB,
                                                    SYNCHRONOUS_FB,
                                                    VIRTUAL);

   type DESTINATION = (LOGFILE, SCREEN);

   type DISABLE_MODE = (CALCULATE, FROZEN);

   type LOGGING_ATTRIBUTE = (CYCLIC, ON_CHANGE);

   type LOGGING_DESTINATION = (ARCHIVE_ONLY, ARCHIVE_AND_DATA_SET);

   type PRIVILEGE = (LOG_PRIVILEGE, WRITE_PRIVILEGE, START_PRIVILEGE);
   -- LOG_PRIVILEGE allows to log, snapshot and trace items in log file
   -- WRITE_PRIVILEGE allows to assign values to model items
   -- START_PRIVILEGE (i.e. session ownership) allows to
   --   * start, stop, and abort the simulation
   --   * load and store simulation state (state vector)
   --   * set minframe interval/increment
   --   * set simulated mission time
   --   * grant and withdraw privileges to/from other users
   -- START_PRIVILEGE it is initially assigned to the user who started CSS_UI
   -- here users are identified by combination of user name and host name

   type SIMULATOR_MODE = (REALTIME, STANDALONE);

   type SIMULATION_STEP = UNSIGNED_INTEGER (0 .. 4294967295);

   type TABLE_ACTION = (ADD, REMOVE);

   type TABLE_CLASS = (MONITORING, LOGGING);

   type TIME_CLASS = (LOCAL, SMT);

   type TIME_MODE = (VIRTUAL, AUTO);

-------------------------------------------------------------------------
-- OPERATIONS
-------------------------------------------------------------------------

  ------------------------
  -- Command execution --
  ------------------------

  ###

  procedure START_CSS_UI      ###
```

Copyright per DIN 34

```
    (in MODEL : CSS_FB_DATABASE_ITEMS;
     in HOST  : STRING := "";
     in MODE  : SIMULATOR_MODE := REALTIME);

  -- Description:
  --   Start the CSS user inteface.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --     HOST  : name of target machine on which to start CSS UI
  --             by default name of local machine running ICP
  --     MODE  : simulation mode (REALTIME or STANDALONE)
  -- Constraints:
   --    at maximum 10 CSS_UI instances can be started in parallel from an ICP
instance
  --   not more than 1 CSS_UI/simulator instance per simulation model at a time can
be started from an ICP instance
  -- Note:
  --   simulation mode REALTIME is more restrictive
   --    * simulation attributes are fixed (i.e. time mode VIRTUAL, factor 1.0
(minframe duration 200 ms), activation mode CONTINUOUS)
   --    * basic simulation control (i.e. start/stop) is done by HLCL commands
START_SMT and STOP_SMT
  --   * various CSS specific HLCL commands are disabled
  -- Example:
  --   START_CSS_UI \EURECA\SIMULATOR\TEST\MODEL_1, "ws_node_1"

  procedure STOP_CSS_UI       ###
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Stop the CSS user interface.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI instance to be affected
   --                 can be omitted if only one CSS_UI instance started from ICP
instance
  -- Constraints:
  --   CSS_UI on given simulation model started

  procedure IS_CSS_UI_STARTED ###
    (in  MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \;
     out RESULT : BOOLEAN);

  -- Description:
  --   Check whether CSS_UI has been started on given simulation model.
  -- Parameters:
```

Copyright per DIN 34

```
  --   IN:
  --     MODEL  : pathname of simulation model
  --                can be omitted if only one, if any, CSS_UI instance started from
ICP instance
  --   OUT:
  --     RESULT : true if CSS_UI started, otherwise false
  -- Constraints:
  --   none

  ###

  procedure SET_SIMULATION_ATTRIBUTES
    (in ACTIVATION : ACTIVATION;
     in STEPS      : SIMULATION_STEP;
     in TIME_MODE  : TIME_MODE;
     in FACTOR     : REAL := 1.0;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
   --     Set simulation attributes applicable to starting simulator kernel (via
START_CSS_KERNEL).
  -- Parameters:
  --   IN:
  --     ACTIVATION : activation mode (STEPWISE, CONTINUOUS)
  --                      in STEPWISE mode the simulation is suspended automatically
after given number of simulation steps (minframes) have been executed
  --                    in CONTINUOUS mode the simulation proceeds until suspended by
STOP_SIMULATION
  --     STEPS      : number of simulation steps (i.e. minframes) to be executed on
START_SIMULATION if activation mode is STEPWISE
  --     TIME_MODE  : time mode (AUTOMATIC, VIRTUAL)
                           in AUTOMATIC mode the simulation is executed as fast as
possible, next simulation step is started as soon as processing of previous step has
finished (dynamic frame durations in LOT)
                         in VIRTUAL mode ths simulation is executed with fixed frame
durations in LOT (static time raster)
                              after processing of simulation step has finished the
simulation pauses until minframe duration expired
  --     FACTOR     : minframe duration in LOT given as multiple of 200 ms (only in
time mode VIRTUAL)
                           minframe duration has impact on duration of higher order
synchronous frames given as multiples of the minframe
                       might want to adapt also minframe increment (see SET_MINFRAME)
  --     MODEL      :
  -- Constraints:
  --   CSS_UI on given simulation model started in simulation mode STANDALONE
  -    user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
```

Copyright per DIN 34

```
  --   simulation state loaded
  --   simulation suspended
  --   minimum factor 0.25 (minimum minframe duration 50 ms)
  --   minimum steps 1
  --    in virtual mode the simulation model implementation must allow processing of
all frames in time in worst case

  procedure SET_MINFRAME
    (in DURATION : DURATION;
     in MODEL    : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Set minframe increment, i.e. the amount the SMT is incremented per simulation
step (minframe).
  -- Parameters:
  --   IN:
  --     DURATION : SMT increment
  --     MODEL    :
  -- Constraints:
   --     CSS_UI/simulator on given simulation model started in simulation mode
STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state not yet loaded or simulation initially suspended
  -- Note:
  --   the minframe increment does not affect the minframe duration in LOT
  -- Example:
  --   SET_MINFRAME 5.0 [s]

  procedure SET_TIME
    (in TIME  : TIME;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Set initial value for SMT.
  -- Parameters:
  --   IN:
  --     TIME  : initial value for SMT.
  --     MODEL :
  -- Constraints:
   --     CSS_UI/simulator on given simulation model started in simulation mode
STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state not yet loaded or simulation initially suspended
  -- Example:
  --   SET_TIME 24.12.1996 13:27:55.50
```

Copyright per DIN 34

```
  ###

  procedure START_CSS_KERNEL
    (in KERNEL_HOST : STRING;
     in CMAS_HOST   : STRING;
     in MODEL       : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Start simulator kernel (and possibly CMAS) on given host(s).
  -- Parameters:
  --   IN:
  --     KERNEL_HOST : name of host to start simulator kernel on
  --     CMAS_HOST   : name of host to start CMAS on
  --                   empty string if kernel should be started without CMAS
  --     MODEL       : pathname of simulation model
  --                   denotes CSS_UI/simulator instance to be affected
  --                    can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI on given simulation model started
  --   simulator kernel not yet connected
  -- Example:
  --   START_CSS_KERNEL ”sim_node_1”, ””

  procedure STOP_CSS_KERNEL
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Stop simulator kernel.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --    simulation state not yet loaded or simulation suspended or simulation faulty
halted

  procedure CONNECT_TO_CSS_KERNEL
    (in HOST  : STRING := ””;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Connect to a running simulator kernel.
```

Copyright per DIN 34

```
  -- Parameters:
  --   IN:
  --     HOST  : name of host running simulator kernel
  --             by default local host running ICP
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running but not yet connected

  procedure DISCONNECT_FROM_CSS_KERNEL
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Disconnect from a running simulator kernel (leaving
  --   it up and running).
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected

  procedure IS_CONNECTED_TO_KERNEL  ###
    (in  HOST   : STRING := "";
     in  MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \;
     out RESULT : BOOLEAN);

  -- Description:
  --   Check whether CSS_UI is connected to simulator kernel on given host.
  -- Parameters:
  --   IN:
  --     HOST   : name of host running simulator kernel
  --     MODEL  : pathname of simulation model
  --              denotes CSS_UI/simulator instance to be affected
  --               can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  --   OUT:
  --     RESULT : true if connected, otherwise false
  -- Constraints:
  --   none

  ###
```

Copyright per DIN 34

```
  procedure LOAD
    (in VECTOR : STRING;
     in MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Load simulation state (state vector).
  -- Parameters:
  --   IN:
  --     VECTOR : name of simulation state (state vector) to be loaded
  --     MODEL  : pathname of simulation model
  --              denotes CSS_UI/simulator instance to be affected
  --               can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state not yet loaded or simulation suspended
  --   given simulation state (state vector) existing

  procedure STORE
    (in VECTOR : STRING;
     in MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Store simulation state (state vector).
  -- Parameters:
  --   IN:
  --     VECTOR : name of simulation state (state vector)
  --     MODEL  : pathname of simulation model
  --              denotes CSS_UI/simulator instance to be affected
  --               can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation suspended or faulty halted
  --   name must comply with Ada identifier syntax, at maximum 16 characters

  ###

  procedure START_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
```

Copyright per DIN 34

```
  --   Start or continue a simulation.
    --       Behaviour depends on current simulation attributes as set by
SET_SIMULATION_ATTRIBUTES
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
   --     CSS_UI/simulator on given simulation model started in simulation mode
STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation suspended
  -- Note:
   --    in simulation mode REALTIME the simulation is controlled by HLCL commands
START_SMT and STOP_SMT.

  procedure STOP_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Suspend a running simulation, halt SMT.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
   --     CSS_UI/simulator on given simulation model started in simulation mode
STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation running

  procedure ABORT_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Abort simulation in faulty state.
   --    After aborting a simulation the user can only store a state vector or
shut–down the simulator.
  -- Parameters:
  --   IN:
```

Copyright per DIN 34

```
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                    can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
   --     CSS_UI/simulator on given simulation model started in simulation mode
STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation running faulty

  ###

  procedure SET_ASSIGN
    (in DISABLE : DISABLE_MODE := CALCULATE;
     in TIME    : union (TIME, DURATION) := 0.0 [s];
     in CLASS   : TIME_CLASS := SMT);

  -- Description:
  --   Set attributes applicable to assignments to simulation model subitems.
  -- Parameters:
  --   IN:
  --      DISABLE : configure assignments to be persistent (perform implicit FREEZE)
or not
   --                  by default assignments will not be persistent (i.e. assigned
value will be overridden by next assignment in function block implementation when
executed)
  --     TIME    : configure execution of assignments
  --               immediate execution   : 0.0 [s]
   --                    time tagged execution : either at fixed time or with delay
relative to time of assignment
  --               by default assignments will be executed immediately
  --     CLASS   : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   none
  -- Example:
  --   SET_ASSIGN CALCULATE, 100.0, LOCAL
  --   \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.OUTPUT = 305.76
   --    -- value of variable BLOCK_A.OUTPUT will be set to 305.76 with delay of 100
sec in local time
  --   SET_ASSIGN FROZEN
  --   \EURECA\TEST\MODEL_1\BOOM1\VALVE3.OUTPUT = 0.0
  --   -- value of variable VALVE3.OUTPUT will be set to 0.0 and frozen immediately,
i.e. value will stay 0.0 until released via REACTIVATE

  procedure FREEZE
    (in SUBITEM_PATH : pathname.*;
```

Copyright per DIN 34

```
     in TIME         : union (TIME, DURATION) := 0.0 [s];
     in CLASS        : TIME_CLASS := SMT);

  -- Description:
  --   Freeze value of simulation model subitem.
  --   Lock subitem by disabling updates from
  --     * internal: assignments in simulation model implementation (atomic function
blocks)
   --       * external: assignments via CSS_UI or HLCL and updates from CMAS, if
connected
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     TIME         : immediate execution   : 0.0 [s]
  --                      time tagged execution : either at fixed time or with delay
relative to current time
  --                    by default the command will be executed immediately
  --     CLASS        : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs WRITE_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
   --    subitem one of simulation model top level input, atomic function block
output, parameter output, composite function block output but not of pulse or burst
pulse type

  procedure REACTIVATE
    (in SUBITEM_PATH : pathname.*;
     in TIME         : union (TIME, DURATION) := 0.0 [s];
     in CLASS        : TIME_CLASS             := SMT);

  -- Description:
  --   Revert effects of FREEZE command.
  --   Release lock on simulation model subitem by enabling updates from
  --     * internal: assignments in simulation model implementation (atomic function
blocks)
   --       * external: assignments via CSS_UI or HLCL and updates from CMAS, if
connected
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     TIME         : immediate execution   : 0.0 [s]
  --                      time tagged execution : either at fixed time or with delay
relative to current time
  --                    by default the command will be executed immediately
```

Copyright per DIN 34

```
  --     CLASS        : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs WRITE_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
   --    subitem one of simulation model top level input, atomic function block
output, parameter output, composite function block output but not of pulse or burst
pulse type

  procedure CANCEL_PENDING_COMMANDS
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Cancel all pending (outstanding time tagged) commands issued by user.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --               can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  --   there must be pending commands, all pending commands must be cancelable

  ###

  procedure ACTIVATE
    (in DEFINITION : TABLE_CLASS := LOGGING;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Enable monitoring or logging.
  -- Parameters:
  --   IN:
  --     DEFINITION : kind of operation to be enabled (MONITORING, LOGGING)
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                    can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if definition is LOGGING
  --   simulator kernel running and connected
  --   simulation state loaded

  procedure DEACTIVATE
```

Copyright per DIN 34

```
    (in DEFINITION : TABLE_CLASS := LOGGING;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Disable monitoring or logging.
  -- Parameters:
  --   IN:
  --     DEFINITION : kind of operation to be disabled (MONITORING, LOGGING)
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                    can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if definition is LOGGING
  --   simulator kernel running and connected
  --   simulation state loaded

  procedure LOAD_TABLE
    (in TABLE : STRING;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
   --    Load given simulation table containing monitoring, logging and/or tracing
definitions.
  -- Parameters:
  --   IN:
  --     TABLE : name of simulation table
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table existing
  --   (name must comply with Ada identifier syntax, at maximum 16 characters)

  procedure UNMONITOR
    (in TABLE     : STRING;
     in ITEM_PATH : pathname.*);

  -- Description:
   --    Remove monitoring elements on given simulation model subitem from given
simulation table by removing monitoring definition from simulation table.
  -- Parameters:
  --   IN:
```

Copyright per DIN 34

```
  --     TABLE     : name of simulation table to be updated
  --     ITEM_PATH : pathname of simulation model subitem
  --                 implicitely denotes CSS_UI/simulator instance to be affected
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --   subitem one of simulation model top level input/output, atomic function block
input/output, parameter output, composite function block output already monitored in
given table

  procedure LOG_ITEM
    (in ACTION       : TABLE_ACTION;
     in TABLE        : STRING;
     in SUBITEM_PATH : pathname.*;
     in ATTRIBUTE    : LOGGING_ATTRIBUTE   := CYCLIC;
     in DESTINATION  : LOGGING_DESTINATION := ARCHIVE_ONLY);

  -- Description:
    --     Register/unregister  given  simulation  model  subitem  to  be  logged  by
adding/removing logging definition to/from given simulation table.
  -- Parameters:
  --   IN:
  --     ACTION       : kind of action to be performed (ADD or REMOVE)
  --     TABLE        : name of simulation table to be updated
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     ATTRIBUTE    : logging mode (CYCLIC, ON_CHANGE)
  --     DESTINATION  : logging destination(s) (ARCHIVE_ONLY, ARCHIVE_AND_DATA_SET)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs LOG_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --    if action = ADD     : subitem one of simulation model top level input/output,
atomic  function  block  input/output,  parameter  output,  composite  function  block
output
  --                         equivalent subitem not already logged
  --                          subitem not of VECTOR, MATRIX, COMPLEX, DURATION or TIME
type if destination = ARCHIVE_AND_DATA_SET
  --   if action = REMOVE : equivalent subitem already logged in given table

  procedure TRACE
    (in ACTION      : TABLE_ACTION;
     in TABLE       : STRING;
     in BLOCK_PATH  : CSS_FB_DATABASE_ITEMS;
```

Copyright per DIN 34

```
     in DESTINATION : DESTINATION := SCREEN);

  -- Description:
   --     Register/unregister given atomic function block to be traced (execution
tracing) by adding/removing tracing definition to/from given simulation table.
  -- Parameters:
  --   IN:
  --     ACTION      : kind of action to be performed (ADD or REMOVE)
  --     TABLE       : name of simulation table to be updated
  --     BLOCK_PATH  : pathname of atomic function block
  --                   implicitely denotes CSS_UI/simulator instance to be affected
   --      DESTINATION : output destination, i.e. SCREEN (CSS_UI console window) or
LOGFILE
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if destination is LOGFILE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --   if action = ADD    : atomic function block not already traced to destination
   --    if action = REMOVE : atomic function block already traced to destination in
given simulation table

  procedure SNAPSHOT
    (in SUBITEM_PATH : pathname.*;
     in DESTINATION  : DESTINATION := SCREEN);

  -- Description:
   --     Write current model item value once into logfile or onto screen (CSS_UI
console window).
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
   --      DESTINATION  : output destination, i.e. SCREEN (CSS_UI console window) or
LOGFILE
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs LOG_PRIVILEGE if destination is LOGFILE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input/output, atomic function block
input/output, parameter output, composite function block output
  -- Example:
  --   SNAPSHOT \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.INPUT LOGFILE

  ###
```

Copyright per DIN 34

```
  procedure WAIT
    (in TIME  : union (TIME, DURATION);
     in CLASS : TIME_CLASS := LOCAL;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Suspend command execution until given time or for given duration.
  --   Useful primarily in command sequences.
  -- Parameters:
  --   IN:
  --     TIME  : time to be waited until
  --             duration to be waited for
  --     CLASS : reference system for time (LOCAL, i.e. LOT or SMT)
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --              can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  -- Example:
  --   WAIT 23.08.1995 07:23:17.000
  --   WAIT 100.0, SMT

  ###

  procedure REQUEST
    (in PRIVILEGE : PRIVILEGE;
     in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Request given privilege from session owner.
  -- Parameters:
  --   IN:
  --     PRIVILEGE : privilege requested
  --     MODEL     : pathname of simulation model
  --                 denotes CSS_UI/simulator instance to be affected
  --                   can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected

  procedure GRANT
    (in PRIVILEGE : PRIVILEGE;
     in USER      : STRING;
     in HOST      : STRING;
     in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);
```

Copyright per DIN 34

```
  -- Description:
  --   Grant given privilege to given user on given host.
  -- Parameters:
  --   IN:
  --     PRIVILEGE : privilege to be granted
  --     USER      : user name
  --     HOST      : name of host running user's CSS_UI
  --     MODEL     : pathname of simulation model
  --                 denotes CSS_UI/simulator instance to be affected
  --                   can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected

  procedure WITHDRAW
    (in PRIVILEGE : PRIVILEGE;
     in USER      : STRING;
     in HOST      : STRING;
     in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Withdraw given (previously granted) privilege for given user on given host.
  -- Parameters:
  --   IN:
  --     PRIVILEGE : privilege to be be withdrawn
  --     USER      : user name
  --     HOST      : name of host running user's CSS_UI
  --     MODEL     : pathname of simulation model
  --                 denotes CSS_UI/simulator instance to be affected
  --                   can be omitted if only one CSS_UI/simulator instance started
from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected

  procedure BROADCAST
    (in MESSAGE : STRING;
     in USER    : STRING := "*";
     in MODEL   : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Send message to simulation user(s).
  -- Parameters:
  --   IN:
```

Copyright per DIN 34

```
  --     MESSAGE : message text
  --     USER    : name of user to send message to
  --               by default message is send to all users
  --     MODEL   : pathname of simulation model
  --               denotes CSS_UI/simulator instance to be affected
  --               can be omitted if only one CSS_UI/simulator instance started from
ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  -- Note:
   --    The message is automatically sent to all users. That does not depend on
whether a user name is given or not.      # TODO

  ###

  builtin write access to simulation model subitem

  -- Description:
  --   Assign value to model subitem applying assignment attributes as specified via
SET_ASSIGN.
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs WRITE_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
   --    subitem one of simulation model top level input, atomic function block
output, parameter output, composite function block output
  --   --if subitem of pulse type value must be TRUE
  --   subitems of PULSE resp. BURST_PULSE type must be assigned via TRIGGER command
  --   if subitem of burst pulse type value must be > 0
  -- Example:
  --   \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.UNSIGNED_INTEGER_OUTPUT := 42
  --   trigger \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.PULSE_INPUT
  --   trigger \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.BURST_PULSE_INPUT, 3

  builtin read access to simulation model subitem

  -- Description:
  --   Get value of model subitem.
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input/output, atomic function block
input/output, parameter output, composite function block input/output
  -- Example:
```

Copyright per DIN 34

```
--   var x : unsigned_integer
--   x := \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.UNSIGNED_INTEGER_OUTPUT

end HLCL_CSS_CMDS;
```

The following commands are already described in NO TAG, but they are only used in connection with the CSS features.

TRIGGER

Function: Triggers a PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.

Constraints: Used only with PULSE objects.

TRIGGER

Function: Triggers a BURST_PULSE object.

Parameters:	item	Mode:	Mandatory
		Type:	pathname
		Meaning:	Path to the desired item including the item name.
	increment	Mode:	Optional
		Type:	integer
		Default:	1
		Meaning:	Determines the number of pulses emitted.

Constraints: Used only with BURST_PULSE objects.

H-4 UCL System Libraries

The UCL System Libraries contain high–level real–time procedures for monitoring, control and special test purposes. They are mentioned here to give a better impression of the test and operations capabilities because – as mentioned earlier – all UCL functions are available on HLCL level, too (note that you have to use the IMPORT command first to specify and load the desired library).

For detailed information about the UCL libraries refer to Appendix I.

H-4.1 System Libraries

HLCL allows to inherit complete libraries of UCL procedures and functions. UCL provides an Onboard System Library and a Ground System Library. The library functions and procedures can be called from within HLCL command window or HLCL sequences after an import of the resp. library has been done:

* IMPORT <library pathname>

Copyright per DIN 34

For a list of procedures/functions defined in the library refer to UCL Reference Manual. (see reference in chapter 2)

Copyright per DIN 34

I UCL SYSTEM LIBRARIES

The next chapter contains the UCL system library specification for following libraries:

GROUND_COMMON

The GROUND_COMMON UCL library does not provide any routine – only type and constant declarations

GROUND_LIBRARY

The GROUND_library contains standard ground commands used for control of the ground system.

GRD_CONVERSION

The GRD_CONVERSION library contains all standard procedures used for conversion of items within the Ground Automated procedures / Sequences

GROUND_TO_OB_LIB

This library contains commands sent to the COF DMS system.

GROUND_VALUES

The GROUND_VALUES library contains standard ground commands used for access to measurement values, software variables and derived values.

MATH_LIB

The MATH_LIB library contains standard mathematical operations for real type parameter.

MATH_LIB_LONG

The MATH_LIB_LONG library contains standard mathematical operations for long_real type parameter

MESSAGE_LIBRARY

The MESSAGE_LIBRARY library contains procedures to access system messages.

TC_CONSTRUCTION

The TC CONSTRUCTION library contains operations to construct a Telecommand online in UCL.

RAW_DATA_LIBRARY

The RAW_DATA library contains operations to access a raw data packet

PACKET_LIBRARY

The PACKET library contains operations to access a raw data packet

ATV_LIBRARY

The library ATV_library contains operations to issue nested telecommands, to convert unsegmented time and to handle the sequence counter of TC’s

TEXT_FILE_IO

Copyright per DIN 34

The library TEXT_FILE_IO contains operations for input and output to text files.

BINARY_FILE_IO

The library BINARY_FILE_IO contains operations for input and output to bimary files.

SYSTEM

The library SYSTEM contains some Operation System specific operations.

CALIBRATION_LIB

The library CALIBRATION_LIB contains procedures used for access to calibration description of measurement values. This library replace the calibration part of the ground_values library.

Additional this chapter contains the description of HLCL libraries:

HLCL_CSS_CMDS

The library HLCL_CSS_CMDS contains procedures to define CSS specific prededfined HLCL commands.

HLCL_HCI_CMDS

The library HLCL_HCI_CMDS contains procedures to define HCI specific prededfined HLCL commands.

Copyright per DIN 34

I-1 UCL System library: GROUND_COMMON

The GROUND_COMMON UCL **library** does not provide any routine – only **type** and constant declarations

I-1.1 UCL System library Specification GROUND_COMMON

```
------------------------------------------------------------------------
--
-- ****************************************
--  GROUND_COMMON UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Contains common definitions for all ground system libraries
--
--     Must be compiled for ground with Body Id = 254
--
--     Must be inserted into MDB with Nickname: "GROUND_COMMON"
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : GROUND_COMMON System Library
--      CGS CM       : $Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/ground_common_.ucl#11 $
--*----------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

  type USER_MESSAGE = STRING (255);

  type MSG_GROUP = STRING(4);
  type MSG_TYPE  = STRING(4);

  type VALUE_STRING = STRING (255);

  type ACQUISITION_COLLECTION =
         PATHNAME (EGSE_INTEGER_MEASUREMENT,
                   UNSIGNED_INTEGER_MEASUREMENT,
                   EGSE_FLOAT_MEASUREMENT,
                   DOUBLE_FLOAT_MEASUREMENT,
                   EGSE_DISCRETE_MEASUREMENT,
                   EGSE_BYTE_STREAM_MEASUREMENT,
                   EGSE_INTEGER_SW_VARIABLE,
                   UNSIGNED_INTEGER_SW_VARIABLE,
                   EGSE_FLOAT_SW_VARIABLE,
                   DOUBLE_FLOAT_SW_VARIABLE,
                   EGSE_DISCRETE_SW_VARIABLE,
                   EGSE_BYTE_STREAM_SW_VARIABLE,
                   EGSE_INTEGER_DERIVED_VALUE,
                   UNSIGNED_INT_DERIVED_VALUE,
                   EGSE_FLOAT_DERIVED_VALUE,
                   DOUBLE_FLOAT_DERIVED_VALUE,
                   EGSE_DISCRETE_DERIVED_VALUE,
                   EGSE_STRING_DERIVED_VALUE,
                   EGSE_MONITOR_LIST,
                   PUS_STRUCTURE_ID,
                   ADU_DESCRIPTION,
                   CDU,
                   VIRTUAL);

  type ADU_NAME = PATHNAME (ADU_DESCRIPTION);
  -- Remark:
  --   ADU_DESCRIPTION covers the MDB enditem types
  --   CCSDS_ADU_DESCRIPTION, UNSTRUCTURED_ADU_DESCRIPTION,
  --   STRUCTURED_ADU_DESCRIPTION and PUS_ADU_DESCRIPTION
  --   (used with this meaning also in other type definitions).


  type ADU_STATUS =
    (ADU_NOT_MAINTAINED,  -- not maintained on this testnode
     ADU_NOT_ACQUIRED,    -- maintained, but not acquired
                          -- (start_acquisition not performed)
     ADU_NOT_RECEIVED,    -- acquired, but not received
     ADU_IS_RECEIVED,     -- acquired and received
     ADU_IS_INTERRUPTED,  -- received, but out of sequence
     ADU_IS_STATIC);      -- set by SAS to static (acquired, but not currently
                          -- received)
```

Copyright per DIN 34

```
type ON_OFF = (ON, OFF);


type ROUTE_SAS_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 EGSE_PREDEFINED_TC,
                 PUS_TC,
                 ADU_DESCRIPTION,
                 CDU,
                 VIRTUAL);

type CCSDS_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_PREDEFINED_TC,
                 PUS_TC,
                 PUS_ADU_DESCRIPTION,
                 ADU_DESCRIPTION,
                 CDU,
                 VIRTUAL);

type AP_STATE =
                    -- STATES:
  (AP_RUNNING,      --   running   – execution of instructions
   AP_SUSPENDED,    --   suspended – stopped the execution of instructions
                    --               until resume
   AP_UNKNOWN,      --   not running, not suspended

                    -- RESULTS:
   AP_SUCCESS,      --   finished with success
   AP_FAILURE,      --   finished with error
   AP_ABORTED);     --   terminated on user request

type AP_EXECUTION_STATE =
       AP_STATE (AP_RUNNING .. AP_UNKNOWN);

type AP_COMPLETION_STATE =
  AP_STATE (AP_UNKNOWN .. AP_ABORTED);


type ARCHIVE_COLLECTION =
  (ALL_PACKETS, TM_PACKETS, TC_PACKETS);


type CCSDS_APID_RANGE = INTEGER ( 0 .. 2**12 – 1);
type OLD_APID_RANGE   = INTEGER (–1 .. 2**12 – 1);
-- Remark:
--   The range of CCSDS_APID_RANGE and OLD_APID_RANGE is depend on
--   CGS configuration parameter
--   TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID
--   during runtime (for more information – see $CGS_HOME/etc/configuration.xml).

type AP_ID = INTEGER;
type AP_ID_LIST = ARRAY (1..40) OF AP_ID;

type AP_NAME = PATHNAME (UCL_AUTOMATED_PROCEDURE);

type APPLICATION_ERROR_MESSAGE = USER_MESSAGE;

type APPLICATION_ID = INTEGER;

type APPLICATION_NAME = STRING (20);

type APPLICATION_STATE =
       (SAS_UNLOADED, SAS_RESET, SAS_INIT, SAS_RUNNING, SAS_ABORTED);
-- Remark:
--
--    ------------                    (crash)
--   |SAS_ABORTED | – < ---------------- < -------------- < -------
--    ------------           ^                 ^                ^
--        |                  |        reset      |                |
--        |unload            |    ---------------------- < -------
--        |                  |   |               ^                |
--        v                  |   v               |                |
--    ------------         ----------         --------         -----------
--   |SAS_UNLOADED| – > – |SAS_RESET| – > – |SAS_INIT| – > – |SAS_RUNNING|
--    ------------  load   ----------  init   --------  start  -----------
--        ^                  |                 |                |
--        |                  v                 v                |
--         ---------- < ---------------- < ------------------------
--                                 unload

type BIT_COUNT = INTEGER (1 .. 32);

type DOWNLOAD_ITEM_NAME = PATHNAME (EGSE_SOFTWARE);
```

Copyright per DIN 34

```
type ISSUE_ITEM_LIST =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 EGSE_PREDEFINED_TC,
                 CDU,
                 VIRTUAL);


type TC_ITEM_LIST =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 EGSE_PREDEFINED_TC,
                 SWOP_COMMAND,
                 CDU,
                 VIRTUAL);


type ISSUE_ITEM_NAME =
       PATHNAME (GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 EGSE_PREDEFINED_TC);

type SINGLE_GDU_NAME =
       PATHNAME (EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 EGSE_PREDEFINED_TC);


type SINGLE_TC_NAME =
       PATHNAME (EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 SWOP_COMMAND,
                 EGSE_PREDEFINED_TC);


type MONITOR_ITEM_NAME =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 UNSIGNED_INTEGER_MEASUREMENT,
                 EGSE_FLOAT_MEASUREMENT,
                 DOUBLE_FLOAT_MEASUREMENT,
                 EGSE_DISCRETE_MEASUREMENT,
                 EGSE_BYTE_STREAM_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 UNSIGNED_INTEGER_SW_VARIABLE,
                 EGSE_FLOAT_SW_VARIABLE,
                 DOUBLE_FLOAT_SW_VARIABLE,
                 EGSE_DISCRETE_SW_VARIABLE,
                 EGSE_BYTE_STREAM_SW_VARIABLE,
                 EGSE_INTEGER_DERIVED_VALUE,
                 UNSIGNED_INT_DERIVED_VALUE,
                 EGSE_FLOAT_DERIVED_VALUE,
                 DOUBLE_FLOAT_DERIVED_VALUE,
                 EGSE_DISCRETE_DERIVED_VALUE,
                 EGSE_STRING_DERIVED_VALUE);

type MONITOR_COLLECTION =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                 UNSIGNED_INTEGER_MEASUREMENT,
                 EGSE_FLOAT_MEASUREMENT,
                 DOUBLE_FLOAT_MEASUREMENT,
                 EGSE_DISCRETE_MEASUREMENT,
                 EGSE_BYTE_STREAM_MEASUREMENT,
                 EGSE_INTEGER_SW_VARIABLE,
                 UNSIGNED_INTEGER_SW_VARIABLE,
                 EGSE_FLOAT_SW_VARIABLE,
                 DOUBLE_FLOAT_SW_VARIABLE,
                 EGSE_DISCRETE_SW_VARIABLE,
                 EGSE_BYTE_STREAM_SW_VARIABLE,
                 EGSE_INTEGER_DERIVED_VALUE,
                 UNSIGNED_INT_DERIVED_VALUE,
                 EGSE_FLOAT_DERIVED_VALUE,
                 DOUBLE_FLOAT_DERIVED_VALUE,
                 EGSE_DISCRETE_DERIVED_VALUE,
                 EGSE_STRING_DERIVED_VALUE,
                 EGSE_MONITOR_LIST,
                 PUS_STRUCTURE_ID,
                 CDU,
                 VIRTUAL);
```

Copyright per DIN 34

```
type ARCHIVE_ITEM_NAME =
       PATHNAME (ADU_DESCRIPTION,
                 GDU_DESCRIPTION_LIST,
                 EGSE_ANALOG_STIMULUS,
                 EGSE_DISCRETE_STIMULUS,
                 EGSE_BINARY_PACKET,
                 PUS_TC,
                 EGSE_PREDEFINED_TC,
                 RESPONSE_PACKET,
                 SWOP_COMMAND,
                 CDU,
                 VIRTUAL);

type FLOAT_ENDITEM=
      PATHNAME (EGSE_FLOAT_MEASUREMENT,
                EGSE_ANALOG_STIMULUS);

type DOUBLE_FLOAT_ENDITEM=
      PATHNAME (DOUBLE_FLOAT_MEASUREMENT);

type INTEGER_ENDITEM =
      PATHNAME (EGSE_INTEGER_MEASUREMENT);

type UNSIGNED_INTEGER_ENDITEM =
      PATHNAME (UNSIGNED_INTEGER_MEASUREMENT);


type DISCRETE_ENDITEM=
      PATHNAME (EGSE_DISCRETE_MEASUREMENT,
                EGSE_DISCRETE_STIMULUS);

type ANALOG_MONITOR_ITEM_NAME =
      PATHNAME (EGSE_INTEGER_MEASUREMENT,
                UNSIGNED_INTEGER_MEASUREMENT,
                EGSE_FLOAT_MEASUREMENT,
                DOUBLE_FLOAT_MEASUREMENT,
                EGSE_INTEGER_SW_VARIABLE,
                UNSIGNED_INTEGER_SW_VARIABLE,
                EGSE_FLOAT_SW_VARIABLE,
                DOUBLE_FLOAT_SW_VARIABLE,
                EGSE_INTEGER_DERIVED_VALUE,
                UNSIGNED_INT_DERIVED_VALUE,
                DOUBLE_FLOAT_DERIVED_VALUE,
                EGSE_FLOAT_DERIVED_VALUE);

type INTEGER_MONITOR_ITEM_NAME =
      PATHNAME (EGSE_INTEGER_MEASUREMENT,
                EGSE_INTEGER_SW_VARIABLE,
                EGSE_INTEGER_DERIVED_VALUE);


type FLOAT_MONITOR_ITEM_NAME =
       PATHNAME (EGSE_FLOAT_MEASUREMENT,
                 EGSE_FLOAT_SW_VARIABLE,
                 EGSE_FLOAT_DERIVED_VALUE);


type UNSIGNED_MONITOR_ITEM_NAME =
      PATHNAME (UNSIGNED_INTEGER_MEASUREMENT,
                UNSIGNED_INTEGER_SW_VARIABLE,
                UNSIGNED_INT_DERIVED_VALUE);


type DOUBLE_FLOAT_MONITOR_ITEM_NAME =
       PATHNAME (DOUBLE_FLOAT_MEASUREMENT,
                 DOUBLE_FLOAT_SW_VARIABLE,
                 DOUBLE_FLOAT_DERIVED_VALUE);

type BYTE_STREAM_MONITOR_ITEM_NAME =
      PATHNAME (EGSE_BYTE_STREAM_MEASUREMENT,
                EGSE_BYTE_STREAM_SW_VARIABLE,
                EGSE_STRING_DERIVED_VALUE);

type DISCRETE_MONITOR_ITEM_NAME =
      PATHNAME (EGSE_DISCRETE_MEASUREMENT,
                EGSE_DISCRETE_SW_VARIABLE,
                EGSE_DISCRETE_DERIVED_VALUE);

type ENDITEM_WITH_RAW_VALUE =
      PATHNAME (EGSE_INTEGER_MEASUREMENT,
                EGSE_FLOAT_MEASUREMENT,
                EGSE_DISCRETE_MEASUREMENT,
                DOUBLE_FLOAT_MEASUREMENT,
                UNSIGNED_INTEGER_MEASUREMENT,
                EGSE_BYTE_STREAM_MEASUREMENT
                );

type ENDITEM_WITH_UNSIGNED_RAW_VALUE =
      PATHNAME (EGSE_INTEGER_MEASUREMENT,
                EGSE_FLOAT_MEASUREMENT,
```

Copyright per DIN 34

```
                DOUBLE_FLOAT_MEASUREMENT,
                UNSIGNED_INTEGER_MEASUREMENT,
                EGSE_DISCRETE_MEASUREMENT);

type ENDITEM_WITH_SIGNED_RAW_VALUE =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                DOUBLE_FLOAT_MEASUREMENT,
                EGSE_FLOAT_MEASUREMENT);

type ENDITEM_WITH_FLOAT_RAW_VALUE =
       PATHNAME (EGSE_FLOAT_MEASUREMENT);

type ENDITEM_WITH_DOUBLE_FLOAT_RAW_VALUE =
       PATHNAME (DOUBLE_FLOAT_MEASUREMENT);

type ENDITEM_WITH_BYTESTREAM_RAW_VALUE =
       PATHNAME (EGSE_BYTE_STREAM_MEASUREMENT);

type ENDITEM_WITH_ENG_VALUE =
       PATHNAME (EGSE_INTEGER_MEASUREMENT,
                EGSE_FLOAT_MEASUREMENT,
                EGSE_DISCRETE_MEASUREMENT,
                EGSE_BYTE_STREAM_MEASUREMENT,
                DOUBLE_FLOAT_MEASUREMENT,
                UNSIGNED_INTEGER_MEASUREMENT,
                EGSE_INTEGER_SW_VARIABLE,
                UNSIGNED_INTEGER_SW_VARIABLE,
                EGSE_FLOAT_SW_VARIABLE,
                DOUBLE_FLOAT_SW_VARIABLE,
                EGSE_DISCRETE_SW_VARIABLE,
                EGSE_BYTE_STREAM_SW_VARIABLE,
                EGSE_INTEGER_DERIVED_VALUE,
                UNSIGNED_INT_DERIVED_VALUE,
                EGSE_FLOAT_DERIVED_VALUE,
                DOUBLE_FLOAT_DERIVED_VALUE,
                EGSE_DISCRETE_DERIVED_VALUE,
                EGSE_STRING_DERIVED_VALUE);

type DISCRETE_VALUE =
       PATHNAME (EGSE_DISCRETE_MEASUREMENT);

type NODE_NAME = PATHNAME (EGSE_NODE);

type USER_MESSAGE_NAME = PATHNAME (EGSE_USER_MESSAGE);

type UCL_ITEM_NAME = PATHNAME (UCL_AUTOMATED_PROCEDURE,
                               UCL_USER_LIBRARY);


type CMD_VERIFICATION_STATUS =
                    -- STATES:
  (STARTED,         --   Verification is started
                    --   (i.e. is in Activation Delay)
   IN_PROGRESS,     --   Verification is in progress
                    --   (i.e. is in Verification Timeout)

                    -- RESULTS:
   TC_VERIF_OK,     --   Verification performed with success
   TC_VERIF_FAILED, --   Verification performed with no success
                    --   (i.e. Verification Timeout elapsed)
   ERROR,           --   Error during Verification Timeout
   NO_VERIFICATION, --   No Verification defined for this enditem
   ABORTED,         --   Verification is aborted due to new issue of same
                    --   command or stop of remote TES
   UNDEFINED,       --   The verifcation status is not defined
                    --   (sending aborted due to error,
                    --    no command sent yet, etc)
   TC_VERIF_DISABLED); --  The verifcation is disabled by user.

type ACQUISITION_STATUS =
    (VALID,         -- the item is valid
     NOT_ACQ,       -- the item is not acquired
     NOT_RECVD,     -- the item hasn't been received after start of acquisition
     NOT_MAINTAINED, -- the item is not known on the test node
     DISAB,         -- processing (calibration) is disabled for the item
     INVALID,       -- the value of the item is invalid
     STATIC);       -- there is no update to the item at the moment


type MONITOR_STATUS =
  (MONITORING_DISABLED,    -- no monitor item(s) or all monitor item(s) disabled
   IN_LIMITS,              -- at least one item in limit, all other in disabled
   OUT_OF_NOMINAL_LIMITS,  -- at least one item out of nominal limit,
                           -- all other in status NONE or IN_NOMINAL_LIMITS
   OUT_OF_DANGER_LIMITS);  -- at least one item out of danger limit, all other in
                           -- status NONE or IN_NOMINAL_LIMITS or OUT_OF_NOMINAL_LIMITS


type LIMIT_SET_NUMBER = INTEGER (0..5);
-- Remark:
--   LIMIT_SET_NUMBER = 0 means the current limit set.
```

Copyright per DIN 34

```
   type BYTE_ARRAY = ARRAY (1..64) of WORD;
   -- Remark:
   --   this are up to 256 bytes

   type COEFFICIENT_RANGE    = INTEGER  (1..5);
   type POINT_PAIR_RANGE     = INTEGER  (1..20);
   type STATECODE_PAIR_RANGE = INTEGER  (0..256);
   -- Remark:
   --   STATECODE_PAIR_RANGE = 0 means UNDEFINED_VALUE statecode

   type RAW_VALUE_TYPE = (R_SIGNED,
                          R_UNSIGNED,
                          R_FLOAT,
                          R_DOUBLE_FLOAT,
                          R_BYTE_STREAM);

   type ENG_VALUE_TYPE = (E_SIGNED,
                          E_UNSIGNED,
                          E_FLOAT,
                          E_DOUBLE_FLOAT,
                          E_STATE_CODE,
                          E_STRING);

   type PICTURE_ID = INTEGER;

   type PICTURE_NAME = PATHNAME (WDU_GROUND_SYNOPTIC_DISPLAY, USS_DISPLAY);

   type PRIORITY = (LOW, HIGH);
   -- Remark:
   --   The elements of this type are overloaded by UCL intrinsics.
   --   Therefor they should noted allways qualified (sorry).
   --   Use Ground_Common.Low and Ground_Common.High !

   type TIME_BASE = (GUARANTEED_LOCAL_TIME, LOCAL_TIME,  SMT);

   type UCL_RETURN       = INTEGER;
   type UCL_STATUS_TEXT  = STRING(120);
   type UCL_STATUS_CODES = ARRAY (1..132) OF UCL_STATUS_TEXT;


-------------------------------------------------------------------------
-- CONSTANTS
-------------------------------------------------------------------------

   constant NULL: CHARACTER := CHARACTER (0);
   constant NULL_APPLICATION_NAME : APPLICATION_NAME  := "";

   constant DEFAULT_ARCH_FILE_OPEN_PERIOD: DURATION := 1800.0 [s];

   constant DEFAULT_ISSUE_TIMEOUT: DURATION    := 0.5 [s];
   constant DEFAULT_ADU_WAIT_TIMEOUT: DURATION := 0.0 [s];
   -- Remark:
   --   DEFAULT_ADU_WAIT_TIMEOUT means: wait forever


   constant DEFAULT_MESSAGE: USER_MESSAGE := "";

   constant OK:                          UCL_RETURN :=   1;
   constant AP_IS_EXECUTING:             UCL_RETURN :=   2;
   constant BUSY:                        UCL_RETURN :=   2;
   constant INVALID_PARAMETER:           UCL_RETURN :=   3;
   constant DISC_FULL:                   UCL_RETURN :=   4;
   constant INVALID_AP_ID:               UCL_RETURN :=   5;
   constant INVALID_AT_TIME:             UCL_RETURN :=   6;
   constant INVALID_CYCLE:               UCL_RETURN :=   7;
   constant INVALID_EXC_COUNT:           UCL_RETURN :=   8;
   constant INVALID_ITEM_NAME:           UCL_RETURN :=   9;
   constant INVALID_LIMIT:               UCL_RETURN :=  10;
   constant INVALID_LIMIT_SET:           UCL_RETURN :=  11;
   constant INVALID_APPLICATION_ID:      UCL_RETURN :=  12;
   constant INVALID_NODE_NAME:           UCL_RETURN :=  13;
   constant INVALID_OFFSET:              UCL_RETURN :=  14;
   constant INVALID_PICTURE_ID:          UCL_RETURN :=  15;
   constant INVALID_PICTURE_NAME:        UCL_RETURN :=  16;
   constant INVALID_SIZE:                UCL_RETURN :=  17;
   constant ITEM_IS_DISABLED:            UCL_RETURN :=  18;
   constant ITEMS_NOT_ACQUIRED:          UCL_RETURN :=  19;
   constant MESSAGE_TOO_BIG:             UCL_RETURN :=  20;
   constant NO_ADU_SERVICE:              UCL_RETURN :=  21;
   constant NO_DELTA_LIMIT:              UCL_RETURN :=  22;
   constant APPLICATION_NACK:            UCL_RETURN :=  23;
   constant INVALID_APPLICATION_NAME:    UCL_RETURN :=  24;
   constant APPLICATION_NOT_READY:       UCL_RETURN :=  25;
   constant TIMEOUT:                     UCL_RETURN :=  26;
   constant TOO_MANY_SYNOPTICS:          UCL_RETURN :=  27;
   constant TOO_MANY_APS:                UCL_RETURN :=  28;
   constant URT_UNKNOWN:                 UCL_RETURN :=  29;
   constant INVALID_ADU_TYPE:            UCL_RETURN :=  30;
   constant INVALID_TESTNODE_MODE:       UCL_RETURN :=  31;
   constant ITEM_NOT_SAS_COMPATIBLE:     UCL_RETURN :=  32;
```

Copyright per DIN 34

```
constant ITEM_NOT_FEE_COMPATIBLE:     UCL_RETURN :=  33;
constant ITEM_NOT_UUT_COMPATIBLE:     UCL_RETURN :=  34;
constant NO_GDU_SERVICE:              UCL_RETURN :=  35;
constant ITEM_IS_ENABLED:             UCL_RETURN :=  36;
constant INVALID_TIME:                UCL_RETURN :=  37;
constant TIME_SERVICE_FAILED:         UCL_RETURN :=  38;
constant OPERATION_ABORTED:           UCL_RETURN :=  39;
constant NOT_MTP:                     UCL_RETURN :=  40;
constant ARCHIVE_FILE_ERROR:          UCL_RETURN :=  41;
constant AP_NOT_COMPILED:             UCL_RETURN :=  42;
constant AP_IS_SUSPENDED:             UCL_RETURN :=  43;
constant ARCHIVING_IS_DISABLED:       UCL_RETURN :=  44;
constant WORKSTATION_NOT_CONNECTED:   UCL_RETURN :=  45;
constant WORKSTATION_NOT_READY:       UCL_RETURN :=  46;
constant INPUT_CANCELLED:             UCL_RETURN :=  47;
constant PROTOCOL_ERROR:              UCL_RETURN :=  48;
constant SYNOPTIC_RESOURCES_EXCEEDED: UCL_RETURN :=  49;
constant CONNECTION_TIMEOUT:          UCL_RETURN :=  50;
constant CONNECTION_ABORTED:          UCL_RETURN :=  51;
constant APPLICATION_LOAD_FAILED:     UCL_RETURN :=  52;
constant APPLICATION_IS_CONNECTED:    UCL_RETURN :=  53;
constant COMMUNICATION_ERROR:         UCL_RETURN :=  54;
constant INVALID_FILE_TYPE:           UCL_RETURN :=  55;
constant INVALID_NODE_TYPE:           UCL_RETURN :=  56;
constant NOT_OK:                      UCL_RETURN :=  57;

-- used in GROUND_TO_OB_LIB
constant SW_COMMAND_NOT_FOUND:        UCL_RETURN :=  58;
constant RESPONSE_TOO_SHORT:          UCL_RETURN :=  59;
constant RESPONSE_TIMEOUT:            UCL_RETURN :=  60;

constant AUTHORIZATION_FAILED:        UCL_RETURN :=  61;
constant PRECONDITION_CHECK_FAILED:   UCL_RETURN :=  62;

-- used in GROUND_TO_OB_LIB
constant INSUFFICIENT_PACKET_SIZE:    UCL_RETURN :=  63;
constant PARAMETER_ERROR:             UCL_RETURN :=  64;

constant MAX_PARALLEL_SWOPS_REACHED:  UCL_RETURN :=  65;

constant RUNTIME_ERROR:               UCL_RETURN := 100;
constant CAL_ERROR :                  UCL_RETURN := 101;
constant ITEM_UNKNOWN :               UCL_RETURN := 102;
constant NOT_YET_IMPLEMENTED :        UCL_RETURN := 103;
constant ACQUISITION_ENABLED :        UCL_RETURN := 104;

constant SESSION_IS_USED :            UCL_RETURN := 110;
constant SESSION_IS_ARCHIVED :        UCL_RETURN := 111;
constant INVALID_SESSION_NAME :       UCL_RETURN := 112;

constant INVALID_TC_HANDLE :          UCL_RETURN := 120;
constant TC_NOT_YET_DEFINED :         UCL_RETURN := 121;
constant NO_HEADER_FOR_BINARY_TC :    UCL_RETURN := 122;
constant NO_PUS_TC :                  UCL_RETURN := 123;
constant NO_FREE_HANDLE :             UCL_RETURN := 124;

constant EXISTING_FILE :              UCL_RETURN := 127;
constant FILE_NOT_FOUND :             UCL_RETURN := 128;
constant FILE_NOT_OPEN :              UCL_RETURN := 129;
constant FILE_DATA_ERROR :            UCL_RETURN := 130;

constant ROUTE_TC_TO_EXTERNAL_OK :    UCL_RETURN := 131;
constant ROUTE_TC_TO_EXTERNAL_NOT_OK: UCL_RETURN := 132;

constant MIN_APPL_ERROR_CODE:         UCL_RETURN := 1000;
constant MAX_APPL_ERROR_CODE:         UCL_RETURN := 9999;
-- Description:
--   reserved range for return code of special application software (SAS)
-- Remark:
--   Applications (SAS) may return their own "ACK-Code" to TES. This code
--   is given to the caller in case of no error situation in the STATUS
--   parameter of each procedure (i.e. instead of OK) or in a separate return
--   parameter.


-- Constants to allow for easy conversion of UCL Status code to readable text

constant UCL_DEFAULT_TEXT: UCL_STATUS_TEXT := "Undefined UCL Status Code";

constant STATUS_STRING: UCL_STATUS_CODES:= (
   "OK",                                                                        -- 1)
   "AP_IS_EXECUTING/BUSY: Function could not be executed: is currently/already executing", -- 2)
   "INVALID_PARAMETER: A parameter is invalid or in conflict with another parameter",      -- 3)
   "DISC_FULL: The disc of the test node or the DB Server is full",             -- 4)
   "INVALID_AP_ID: The ID for the Automated Procedure is invalid / The addressed AP is not running (anymore)", -- 5)
   "INVALID_AT_TIME: The value given for the AT_TIME parameter is invalid",     -- 6)
   "INVALID_CYCLE: The value given for the CYCLE parameter is invalid, range is 1 minute to 23 hours 59 minutes", -- 7)
   "INVALID_EXC_COUNT:The value given for the Exception Count parameter is invalid",       -- 8)
   "INVALID_ITEM_NAME: The name for the item is invalid/not existing/of wrong type",       -- 9)
   "INVALID_LIMIT: The value for the limit is invalid",                        -- 10)
   "INVALID_LIMIT_SET: The limit set number must be in range 0..5",            -- 11)
```

Copyright per DIN 34

```
    "INVALID_APPLICATION_ID: The application id is not known",                          -- 12)
    "INVALID_NODE_NAME: The name for the node is invalid",                              -- 13)
    "INVALID_OFFSET: Offset is not in valid range",                                     -- 14)
    "INVALID_PICTURE_ID",                                                               -- 15)
    "INVALID_PICTURE_NAME",                                                             -- 16)
    "INVALID_SIZE",                                                                     -- 17)
    "ITEM_IS_DISABLED: Function could not be executed, because item is disabled",       -- 18)
    "ITEMS_NOT_ACQUIRED: Function could not be executed, because item is not acquired", -- 19)
    "MESSAGE_TOO_BIG: (e.g.: Parameter List for SWOPs do not fit into CCSDS packet)",   -- 20)
    "NO_ADU_SERVICE: The addressed SAS has not announced its ADU service yet",          -- 21)
    "NO_DELTA_LIMIT",                                                                   -- 22)
    "APPLICATION_NACK: The SAS has given a negative acknowledge",                       -- 23)
    "INVALID_APPLICATION_NAME: SAS name not known / not related / not connected ",      -- 24)
    "APPLICATION_NOT_READY: Something went wrong when preparing/sending a cmd/request to SAS or when receiving the acknowlege",-- 25)
    "TIMEOUT: The acknowlegement of the SAS was not received in time", -- 26)
    "TOO_MANY_SYNOPTICS: The number of synoptics already loaded exceeds the allowable limit", -- 27)
    "TOO_MANY_APS: The number of APs already loaded exceeds the allowable limit",       -- 28)
    "URT_UNKNOWN",                                                                      -- 29)
    "INVALID_ADU_TYPE: The type of the ADU is not compatible with the called function or the ADU is not simulated/acquired",-- 30)
    "INVALID_TESTNODE_MODE: The mode of the testnode (TES) does not allow for this function", -- 31)
    "ITEM_NOT_SAS_COMPATIBLE: Error when downloading file to SAS",                      -- 32)
    "ITEM_NOT_FEE_COMPATIBLE",                                                          -- 33)
    "ITEM_NOT_UUT_COMPATIBLE",                                                          -- 34)
    "NO_GDU_SERVICE:The addressed SAS has not announced its GDU service yet",           -- 35)
    "ITEM_IS_ENABLED: Function could not be executed, because item is enabled",         -- 36)
    "INVALID_TIME: Time parameter or time in CCSDS Header invalid",                     -- 37)
    "TIME_SERVICE_FAILED: The access to the time service failed",                       -- 38)
    "OPERATION_ABORTED: Operation was aborted (e.g. by forced_stop of test node or by abort_ap)", -- 39)
    "NOT_MTP: The addressed test node requires to be the MTP",                          -- 40)
    "ARCHIVE_FILE_ERROR: The access to an archive file failed",                         -- 41)
    "AP_NOT_COMPILED: The addressed AP needs to be compiled in the MDB",                -- 42)
    "AP_IS_SUSPENDED: Function could not be executed, because AP is suspended",        -- 43)
    "ARCHIVING_IS_DISABLED: Function could not be executed, because archiving is (already) disabled", -- 44)
    "WORKSTATION_NOT_CONNECTED: Addressed workstation (HCI) is not connected to the test node", -- 45)
    "WORKSTATION_NOT_READY:Addressed workstation (HCI) is not ready",                   -- 46)
    "INPUT_CANCELLED",                                                                  -- 47)
    "PROTOCOL_ERROR: Something went wrong when communicating with other software / Invalid Response Packet for SWOP", -- 48)
    "SYNOPTIC_RESOURCE_ERROR: to many synoptics open or toolkit not supported",                                 -- 49)
    "CONNECTION_TIMEOUT",                                                               -- 50)
    "CONNECTION_ABORTED",                                                               -- 51)
    "APPLICATION_LOAD_FAILED: The loading of the SAS executable failed",                -- 52)
    "APPLICATION_IS_CONNECTED: The SAS is already connected",                           -- 53)
    "COMMUNICATION_ERROR: Error on the communication channel",                          -- 54)
    "INVALID_FILE_TYPE",                                                                -- 55)
    "INVALID_NODE_TYPE",                                                                -- 56)
    "NOT_OK: For Operations on Lists: Some of the items could not be processed / executed", -- 57)
    "SW_COMMAND_NOT_FOUND: The addressed Software Command was not found",               -- 58)
    "RESPONSE_TOO_SHORT: The response packet for a Software Command did not contain all expected parameters", -- 59)
    "RESPONSE_TIMEOUT: The response packet for a Software Command was not received in time", -- 60)
    "AUTHORIZATION_FAILED: A critical command (GDU) was not authorized",                -- 61)
    "PRECONDITION_CHECK_FAILED: The check of command (GDU) preconditions failed",       -- 62)
    "INSUFFICIENT_PACKET_SIZE:",                                                        -- 63)
    "PARAMETER_ERROR: Parameter List of SWOP has wrong data / wrong types",             -- 64)

    "MAX_PARALLEL_SWOPS_REACHED: too many parallel SWOP commands, increase Config Parameter if necessary",  -- 65)

   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 66..99
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 70..74
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 75..79
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 80..84
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 85..89
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 90..94
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 95..99
    "RUNTIME_ERROR: Something went wrong in the software",                              -- 100
    "CAL_ERROR: Error during  (de-)calibration",                                        -- 101
    "ITEM_UNKNOWN: Referenced item is not known in the test node's local data structures", -- 102
    "NOT_IMPLEMENTED: Addressed procedure/function is not yet implemented in the executing software", -- 103
    "ACQUISITION_ENABLED: Error during modification/access of Calibration Info: Item is in wrong state (acquired)", -- 104
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,   -- 105..109
    "SESSION_IS_USED: Session is used for FA operation or deletion",                    -- 110
    "SESSION_IS_ARCHIVED: Session is archived",                                         -- 111
    "INVALID_SESSION_NAME: The name of the session is not correct",                     -- 112
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,                                   -- 113..115
   UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,UCL_Default_Text,                  -- 116..119
    "INVALID_TC_HANDLE: TC Handle is not in specified range",                           -- 120
    "TC_NOT_YET_DEFINED: TC packet addressed which is not existing",                    -- 121
    "NO_HEADER_FOR_BINARY_TC: No header existing for Binary_Packets",                   -- 122
    "NO_PUS_TC: No PUS Telecommand",                                                    -- 123 -- currently unused
    "NO_FREE_HANDLE: Maximum Number of TCs already reached",                            -- 124
   UCL_Default_Text,UCL_Default_Text,                                                    -- 125..126
    "EXISTING_FILE: File to be created does already exist", -- 127
    "FILE_NOT_FOUND: Specified file cannot be accessed",                                -- 128
    "FILE_NOT_OPEN: Specified file is not open",                                        -- 129
    "FILE_DATA_ERROR: Tried to read text from binary file (or vice versa)",             -- 130
    "ROUTE_TC_TO_EXTERNAL_OK: The TC is not send to destination, it was route to external successfully.",   -- 131
    "ROUTE_TC_TO_EXTERNAL_NOT_OK: The TC is not send to destination, the try to route to external fails."); -- 132

END GROUND_COMMON;
```

Copyright per DIN 34

I-2 UCL System library: GROUND_library

The GROUND_**library** contains standard ground commands used for control of the ground system.

library Id (Body_Id): 2

I-2.1 UCL Ground System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  GROUND_LIBRARY UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to control the execution of test
--
--     Must be compiled with Body Id = 2
--                      for Language = UCL
--                      and for Ground.
--
--     recommended Nickname: ”GROUND_LIBRARY”
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  :  GROUND_LIBRARY System Library
--      CGS CM       : $Id: //cgs/7.3.DEV/src/distribution/gsaf/cgsi/lib/ucl/ground_library_.ucl#2 $
--*----------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library GROUND_LIBRARY;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

  import GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

  type T_SESSION_NAME = STRING (80);

------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  ------------------------------------------------------------------------
  -- Time Management
  ------------------------------------------------------------------------

  1:  function  CLOCK (in BASE: TIME_BASE := LOCAL_TIME) : TIME;

  -- Description:
  --   Returns the actual local time or SMT
  -- Parameters:
  --   IN:
  --     BASE : time_base to use


  2:  procedure DELAY
        (in  DELAY: DURATION;
         in  BASE:  TIME_BASE := LOCAL_TIME);

  -- Description:
  --   Waits the given time interval
  -- Parameters:
  --   IN:
  --      DELAY : duration to delay
  --      BASE  : time_base to use

  3:  procedure WAIT_UNTIL
        (in  DUE_TIME: TIME;
         in  BASE:     TIME_BASE := LOCAL_TIME;
         out STATUS:   UCL_RETURN);

  -- Description:
  --   Waits until the given time is reached
  -- Parameters:
  --   IN:
  --     DUE_TIME : time to wait until
  --     BASE     : time_base to use
  --   OUT:
```

Copyright per DIN 34

```
--     STATUS   : UCL return status
--                   OK
--                   OPERATION_ABORTED
--                   TIME_SERVICE_FAILED
--                   INVALID_TIME

4:  guarded procedure START_SMT
      (in  VALUE:  TIME := ~:~ ;
       out STATUS: UCL_RETURN);

-- Description:
--   Starts Simulated Mission Time (SMT) or set SMT to a given value and start.
--   If SMT is already running, the new VALUE is immediately set, and
--   time is incremented accordingly thereafter.
-- Restriction:
--   Only valid on Master Time Processor (MTP)
-- Parameters:
--   IN:
--     VALUE   : new SMT; by default current SMT is resumed
--   OUT:
--     STATUS  : UCL return status
--                   OK
--                   NOT_MTP
--                   TIME_SERVICE_FAILED

5:  guarded procedure START_SMT_WITH_OFFSET
      (in  OFFSET: INTEGER;
       out STATUS: UCL_RETURN);

-- Description:
--   Starts Simulated Mission Time (SMT) relative to local time. If SMT is
--   already running, the new time acc. to OFFSET is immediately set, and time
--   is incremented accordingly thereafter.
-- Restriction:
--   Only valid on Master Time Processor (MTP)
-- Parameters:
--   IN:
--     OFFSET : number of seconds relative to Local Time (LOT). A negative
--              number means number of seconds before (earlier) than LOT.
--   OUT:
--     STATUS  : UCL return status
--                   OK
--                   NOT_MTP
--                   TIME_SERVICE_FAILED

6:  guarded procedure STOP_SMT (out STATUS: UCL_RETURN);

-- Description:
--   Stop the running Simulated Mission Time.
-- Restriction:
--   Only valid on Master Time Processor (MTP)
-- Parameters:
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   NOT_MTP
--                   TIME_SERVICE_FAILED

7:  guarded procedure SET_SMT
      (in  VALUE:  TIME;
       out STATUS: UCL_RETURN);

-- Description:
--   Sets Simulated Mission Time (SMT) to a given value. If SMT is already
--   running, the new VALUE is immediately set, and time is incremented
--   accordingly thereafter.
-- Restriction:
--   Only valid on Master Time Processor (MTP)
-- Parameters:
--   IN:
--     VALUE  : new SMT
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   NOT_MTP
--                   TIME_SERVICE_FAILED

8:  procedure GET_SMT_STATUS
      (out RUNNING:  BOOLEAN;
       out OFFSET:   INTEGER;
       out STATUS:   UCL_RETURN);

-- Description:
--   Returns the actual SMT status information of the local node.
-- Parameters:
--   OUT:
--     RUNNING : return TRUE  if the SMT is running
--                      FALSE otherwise
--     OFFSET  : return the offset between SMT and LT
--     STATUS  : UCL return status
--               OK :                 ok
```

Copyright per DIN 34

```
--              NOT_OK :              internal problem
--              TIME_SERVICE_FAILED : the time services are not initialized
--                                    correct or not reachable

-------------------------------------------------------------------------
-- AP Handling
-------------------------------------------------------------------------

-- General: The procedures below are available for ground automated procedures

 9:  guarded procedure EXECUTE_AP
       (guarded in  AP:     AP_NAME ();
                in  PRIO:   PRIORITY := GROUND_COMMON.LOW;
                in  NODE:   NODE_NAME := \\;
                out ID:     AP_ID;
                out STATUS: UCL_RETURN);

-- Description:
--   Starts execution of an ground AP. If the procedure is called from within
--   an AP, the called AP runs asynchronously unless SYNCHRONISE_WITH_AP is
--   also called.
-- Restriction:
--   The number of parallel running APs per testnode is restricted up to 40.
-- Parameters:
--   IN:
--     AP   : The name of AP to run including its list of parameters.
--     PRIO : The AP's priority (GROUND_COMMON.HIGH or GROUND_COMMON.LOW).
--     NODE : Node on which the AP is to be executed, by default the current
--            node.
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   TOO_MANY_APS
--                   INVALID_NODE_NAME
--                   AP_NOT_COMPILED
--                   INVALID_PARAMETER

10:  procedure SYNCHRONISE_WITH_AP
      (in  AP:         AP_ID;
       in  NODE:       NODE_NAME :=  \\;
       out COMPLETION: AP_COMPLETION_STATE;
       out STATUS:     UCL_RETURN);

-- Description:
--   Forces the calling AP to synchronise with the called AP. The procedure
--   SYNCHRONISE_WITH_AP blocks until the called AP has finished.
-- Parameters:
--   IN:
--     AP   : internal AP identifier, return value from EXECUTE_AP.
--     NODE : node on which the AP is to be executed, by default the current
--            node
--   OUT:
--     COMPLETION : returns the called AP's completion state.
--                   AP_SUCCESS
--                   AP_FAILURE
--                   AP_ABORTED
--     STATUS     : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   INVALID_AP_ID
--                   INVALID_NODE_NAME

11:  guarded procedure SUSPEND_AP
      (in  AP:     AP_ID;
       in  NODE:   NODE_NAME := \\;
       out STATUS: UCL_RETURN);

-- Description:
--   Suspends execution of a AP on a given test node.
-- Parameters:
--   IN:
--     AP   : internal AP identifier, return value from EXECUTE_AP.
--     NODE : node on which the AP is to be executed, by default the current
--            node
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   INVALID_AP_ID
--                   INVALID_NODE_NAME
--                   INVALID_PARAMETER


12:  guarded procedure RESUME_AP
      (in  AP:     AP_ID;
       in  NODE:   NODE_NAME := \\;
       out STATUS: UCL_RETURN);

-- Description:
--   Resumes execution of a suspended AP
```

Copyright per DIN 34

```
 -- Parameters:
 --   IN:
 --     AP   : internal AP identifier, return value from EXECUTE_AP.
 --     NODE : node on which the AP is to be executed, by default the current
 --            node
 --   OUT:
 --     STATUS : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  INVALID_AP_ID
 --                  INVALID_NODE_NAME
 --                  AP_IS_EXECUTING
 --                  INVALID_PARAMETER

13:  function OWN_AP_ID : AP_ID;

 -- Description:
 --   Returns the AP identifier of the AP calling that function
 -- Restriction:
 --    UCL only – not usable from HLCL

14:  procedure GET_AP_ID
      (in  AP:      AP_NAME;
       in  NODE:    NODE_NAME := \\;
       out NUMBER:  INTEGER;
       out ID_LIST: AP_ID_LIST;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Returns all AP identifiers for a given automated procedure name on a
 --   given test node. Since several instances of a AP may run in parallel
 --   on one test node, a list of AP identifiers is returned
 -- Parameters:
 --   IN:
 --     AP   : name of AP
 --     NODE : node on which the AP is to be executed, by default the current
 --            node
 --   OUT:
 --     NUMBER  : returns the number of AP identifiers in the ID_List for
 --               this AP.
 --     ID_LIST : returns list of AP_IDs found. Unused fields of the ID_List
 --               are filled with zeroes.
 --     STATUS  : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_NODE_NAME
 --                   INVALID_PARAMETER

15:  procedure GET_AP_STATUS
      (in  AP:         AP_ID;
       in  NODE:       NODE_NAME := \\;
       out EXEC_STATE: AP_EXECUTION_STATE;
       out STATUS:     UCL_RETURN);

 -- Description:
 --   Returns the current status of a AP.
 --   For a unkwown AP the UCL status is OK and the EXEC_STATE is AP_UNKNOWN.
 -- Parameters:
 --   IN:
 --     AP   : internal AP identifier, return value from EXECUTE_AP.
 --     NODE : node on which the AP is to be executed, by default the current
 --            node
 --   OUT:
 --     EXEC_STATE : returns the execution state of the AP
 --                  AP_RUNNING
 --                  AP_SUSPENDED
 --                  AP_UNKNOWN
 --     STATUS     : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  INVALID_NODE_NAME
 --                  INVALID_PARAMETER

16:  procedure READ_MESSAGE_FROM_AP
      (in  BLOCK:     BOOLEAN := TRUE;
       out AP:        AP_ID;
       out NODE_NAME: USER_MESSAGE;
       out MESSAGE:   USER_MESSAGE;
       out STATUS:    UCL_RETURN);

 -- Description:
 --   Reads a message asynchronously from any another AP
 -- Restriction:
 --    UCL only – not usable from HLCL
 -- Parameters:
 --   IN:
 --     BLOCK : flag controlling the source of the message, as follows:
 --       TRUE  – waits until a AP really sends a message (by default)
 --       FALSE – reads the next message buffered, or if none, returns an
 --               empty string.
 --   OUT:
```

Copyright per DIN 34

```
--     NODE_NAME : returns string identifying the node on which other AP is
--                 running.
--     MESSAGE   : returns string from the other AP.
--     STATUS    : UCL return status
--                   OK
--                   OPERATION_ABORTED

17:  procedure WRITE_MESSAGE_TO_AP
      (in  AP:      AP_ID;
       in  NODE:    NODE_NAME := \\;
       in  MESSAGE: USER_MESSAGE := DEFAULT_MESSAGE;
       out STATUS:  UCL_RETURN);

-- Description:
--   Sends a message asynchronously from one AP to another, i.e. without
--   waiting for the receiving AP to accept the message. The caller simply
--   sends the message and continues.
-- Restriction:
--    UCL only – not usable from HLCL
-- Parameters:
--   IN:
--     AP      : internal AP identifier, return value from EXECUTE_AP.
--     NODE    : node on which the AP is to be executed, by default the
--               current node
--     MESSAGE : string to send
--   OUT:
--     STATUS  : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   INVALID_NODE_NAME
--                   INVALID_AP_ID


-------------------------------------------------------------------------
-- Application Handling
-------------------------------------------------------------------------

-- Remark:
--
--    ------------                    (crash)
--   |SAS_ABORTED | – < ---------------- < --------------- < -------
--    ------------           ^                  ^                ^
--       |                   |        reset     |                |
--       |unload             |    ---------------------- < -------
--       |                   |   |              ^                |
--       v                   |   v              |                |
--    ------------  ---------  --------  ---------  ------------
--   |SAS_UNLOADED| – > – |SAS_RESET| – > – |SAS_INIT| – > – |SAS_RUNNING|
--    ------------  load   ---------  init   --------  start  ------------
--       ^                  |                  |                |
--       |                  v                  v                |
--        ---------- < ---------------- < -------------------------
--                                    unload

18:  guarded procedure LOAD_APPLICATION
      (in  NAME:   STRING;
       in  NODE:   NODE_NAME := \\;
       in  PARAMS: USER_MESSAGE := DEFAULT_MESSAGE;
       out APPL:   APPLICATION_ID;
       out STATUS: UCL_RETURN);

-- Description:
--   Loads an application on a given node (workstation/testnode).
-- Parameters:
--   IN:
--     NAME   : The name of the SAS executable.
--              1) SAS is an string without '/', the SAS will be searched
--                 in the predefined application directory
--                 $CGS_HOME/gsaf/sas/bin.
--              2) SAS string starts with '/' character, the given
--                 executable will be started.
--              3) SAS string starts with '$' character and the string
--                 contains '/', the given executable will be started after
--                 evaluation of environment.
--     NODE   : The name of the node, where the SAS shall execute. By default
--              the empty pathname denotes the local testnode. The
--              workstations are referenced by their pathname to EGSE_NODE
--              entry in CCU.
--     PARAMS : Parameter string up to 255 characters to pass to the
--              application via command line parameter.
--   OUT:
--     APPL   : application ID to be used for subsequent control operations
--     STATUS : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   INVALID_APPLICATION_NAME
--                   APPLICATION_NACK
--                   TIMEOUT
--                   INVALID_ITEM_NAME
--                   ITEM_IS_ENABLED
--                   OPERATION_ABORTED
```

Copyright per DIN 34

```
19:  guarded procedure UNLOAD_APPLICATION
      (in  APPL:   APPLICATION_ID;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Unloads the specified application
 -- Parameters:
 --   IN:
 --     APPL : application ID as returned from LOAD_APPLICATION
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_APPLICATION_NAME
 --                   INVALID_APPLICATION_ID

20:  guarded procedure INIT_APPLICATION
      (in  APPL:    APPLICATION_ID;
       in  MESSAGE: USER_MESSAGE := DEFAULT_MESSAGE;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Initialises the specified application.
 -- Parameters:
 --   IN:
 --     APPL    : application ID as returned from LOAD_APPLICATION
 --     MESSAGE : string up to 255 characters to send to the application
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   APPLICATION_NACK
 --                   COMMUNICATION_ERROR
 --                   TIMEOUT
 --                   INVALID_APPLICATION_ID

21:  guarded procedure START_APPLICATION
      (in  APPL:   APPLICATION_ID;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Starts the application running.
 -- Parameters:
 --   IN:
 --     APPL : application ID as returned from LOAD_APPLICATION
 --   OUT:
 --     STATUS : UCL return status
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   APPLICATION_NACK
 --                   COMMUNICATION_ERROR
 --                   TIMEOUT
 --                   INVALID_APPLICATION_ID

22:  guarded procedure RESET_APPLICATION
      (in  APPL:   APPLICATION_ID;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Resets the specified application.
 -- Parameters:
 --   IN:
 --     APPL    : application ID as returned from LOAD_APPLICATION
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   APPLICATION_NACK
 --                   COMMUNICATION_ERROR
 --                   TIMEOUT
 --                   INVALID_APPLICATION_ID

23:  procedure GET_APPLICATION_STATUS
      (in  APPL:          APPLICATION_ID;
       out STATE:         APPLICATION_STATE;
       out ERROR_COUNT:   INTEGER;
       out ERROR_MESSAGE: APPLICATION_ERROR_MESSAGE;
       out STATUS:        UCL_RETURN);

 -- Description:
 --   Returns the status of an application, including error status and
 --   statistics.
 -- Parameters:
 --   IN:
 --     APPL    : application ID as returned from LOAD_APPLICATION
 --   OUT:
 --     STATE         : state of application.
 --     ERROR_COUNT   : integer count of errors in the application.
 --     ERROR_MESSAGE : error message from the application.
 --     STATUS        : UCL return status
```

Copyright per DIN 34

```
--                  OK
--                  INVALID_TESTNODE_MODE
--                  APPLICATION_NACK
--                  COMMUNICATION_ERROR
--                  TIMEOUT
--                  INVALID_APPLICATION_ID

24:  procedure READ_MESSAGE_FROM_APPLICATION
      (in  BLOCK:   BOOLEAN := TRUE;
       out APPL:    APPLICATION_ID;
       out MESSAGE: USER_MESSAGE;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Reads a message from any application.
 -- Restriction:
 --    UCL only – not usable from HLCL
 -- Parameters:
 --   IN:
 --     BLOCK : boolean flag, controlling the source of the read:
 --       TRUE  – waits until an application really sends a message
 --       FALSE – reads the next message buffered. If none, returns an empty
 --               string and a null application ID.
 --   OUT:
 --     APPL    : ID of the application sending the message
 --     MESSAGE : a string containing the message
 --     STATUS  : UCL return status
 --                  OK
 --                  OPERATION_ABORTED

25:  guarded procedure WRITE_MESSAGE_TO_APPLICATION
      (in  APPL:    APPLICATION_ID;
       in  MESSAGE: USER_MESSAGE;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Sends a message to an application.
 -- Restriction:
 --    UCL only – not usable from HLCL
 -- Parameters:
 --   IN:
 --     APPL    : application ID as returned from LOAD_APPLICATION
 --     MESSAGE : string of up to 255 characters to be sent.
 --   OUT:
 --     STATUS : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  APPLICATION_NACK
 --                  COMMUNICATION_ERROR
 --                  TIMEOUT
 --                  INVALID_APPLICATION_ID

26:  procedure GET_APPLICATION_NAME
      (in  APPL:   APPLICATION_ID;
       out NAME:   APPLICATION_NAME;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Returns the name of an application
 -- Parameters:
 --   IN:
 --     APPL    : application ID as returned from LOAD_APPLICATION
 --   OUT:
 --     NAME   : the name string of the application
 --     STATUS : UCL return status
 --                  OK
 --                  INVALID_APPLICATION_ID

27:  procedure GET_APPLICATION_ID
      (in  NAME:   APPLICATION_NAME;
       out APPL:   APPLICATION_ID;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Returns the application ID of an application, if the application is
 --   already loaded.
 -- Parameters:
 --   IN:
 --     NAME   : the name string of the application
 --   OUT:
 --     APPL   : application ID as returned from LOAD_APPLICATION
 --     STATUS : UCL return status
 --                  OK
 --                  INVALID_APPLICATION_NAME

 ------------------------------------------------------------------------
 -- Issue of HW Stimuli & Downloading
 ------------------------------------------------------------------------

28:  guarded procedure ISSUE
      (guarded in  ITEM:           ISSUE_ITEM_NAME();
```

Copyright per DIN 34

```
              in  PRIO:                   PRIORITY := GROUND_COMMON.LOW;
              in  AT_TIME:                TIME := ~:~;
              in  BASE:                   TIME_BASE := LOCAL_TIME;
              in  ONBOARD_EXECUTION_TIME: TIME := ~:~;
              in  TIMEOUT:                DURATION := default_issue_timeout;
              out SAS_ACK_CODE:           INTEGER;
              out SAS_ACK_TIME:           TIME;
              out CCSDS_SEQUENCE_COUNT:   INTEGER;
              out STATUS:                 UCL_RETURN);

-- Description:
--   Issues a stimulus or a telecommand (TC) or a list of
--   stimulus/telecommands to an application (SAS).
-- Parameters:
--   IN:
--     ITEM : the database enditem to be issued. It can denote a parameter
--            list, as defined for the enditem.
--            Note:  (1) the enditem may also be maintained by a remote node
--                   (2) the enditem may have additional attributes such as
--                       command verification or command authorization
--                       (specified in the MDB). These attributes are taken
--                       into account before and after sending the generated
--                       TC to the addressed SAS. If specific values for TC
--                       verification are to be set online, the command
--                       ISSUE_AND_VERIFY can be used (see also description
--                       there for behaviour on TC verification)
--     PRIO    : the priority to be used for issuing the item
--               (GROUND_COMMON.LOW or GROUND_COMMON.HIGH)
--     AT_TIME : The time to issue the stimulus from application (SAS)
--               Default: no time.
--
--               The purpose of the AT_TIME parameter shall be the same for
--               all GDU alternatives:
--
--               For MDB enditems of type EGSE_ANALOGUE_STIMULUS,
--               EGSE_DISCRETE_STIMULUS, EGSE_BINARY_PACKET and
--               EGSE_PREDEFINED_TC,  the time value is written as the
--               TIME_TAG into the GDU generated by the command and sent to
--               the SAS. The SAS has to extract the TIME_TAG from the
--               GDU and execute the stimulus at the given time.
--
--               For MDB enditems of type GDU_DESCRIPTION_LIST the AT_TIME
--               value will be written into the time tag field of the GDU
--               header of each stimulus referenced within the GDU list.
--
--     BASE    : specifies if AT_TIME is local time or SMT
--
--     ONBOARD_EXECUTION_TIME : The time to execute the telecommand onboard
--               Default:  no time.
--
--               This parameter works on MDB enditems of type
--               EGSE_PREDEFINED_TC only. ONBOARD_EXECUTION_TIME is evaluated
--               depend on the CCSDS primary header flags for TIME_ID (as
--               defined in MDB) and influences the CCSDS primary header
--               part for TIME_FIELD.
--
--                TIME_ID                          |  TIME_FIELD
--               ----------------------------------|-----------------------------
--                00 (NO_TIME_FIELD)               |  Null-Time (00 00 00 00 00)
--                01 (TIME_OF_PACKET_GENERATION)   |  current SMT (time when
--                                                 |  command is prepared / built)
--                10 (TIME_TAG)                    |  ONBOARD_EXECUTION_TIME
--                11 (UNDEFINED)                   |  Null-Time (00 00 00 00 00)
--
--            Note:
--            (1) If no ONBOARD_EXECUTION _TIME parameter is given, i.e. the
--                default value is applied , the TIME_FIELD shall set to
--                Null-Time (00 00 00 00 00) for TIME_ID = 10.
--
--            (2) For MDB enditems of type GDU_DESCRIPTION_LIST the
--                ONBOARD_EXECUTION _TIME value will evaluated for
--                each EGSE_PREDEFINED_TC referenced within the GDU list.
--
--     TIMEOUT : the maximum time to wait for acknowledgement of this
--               stimulus or telecommand
--   OUT:
--     SAS_ACK_CODE : the return code received from SAS
--                    range (0, [ 1000 .. 2**16 - 1 ])
--     SAS_ACK_TIME : the time received from SAS
--     CCSDS_SEQUENCE_COUNT : the value used as sequence counter in a issued
--                     CCSDS Packet (single CCSDS packats only)
--     STATUS : UCL return status
--                    OK
--                    INVALID_TESTNODE_MODE
--                    NO_GDU_SERVICE
--                    APPLICATION_NACK
--                    TIMEOUT
--                    ITEM_IS_DISABLED
--                    APPLICATION_NOT_READY
--                    INVALID_PARAMETER
--                    PRECONDITION_CHECK_FAILED
```

Copyright per DIN 34

```
--                  AUTHORIZATION_FAILED


29:  guarded procedure ISSUE_AND_VERIFY
      (guarded in ITEM:                   SINGLE_GDU_NAME();
               in PRIO:                   PRIORITY := GROUND_COMMON.LOW;
               in AT_TIME:                TIME := ~:~;
               in BASE:                   TIME_BASE := LOCAL_TIME;
               in ONBOARD_EXECUTION_TIME: TIME := ~:~;
               in TIMEOUT:                DURATION := DEFAULT_ISSUE_TIMEOUT;
               in ACTIVATION_DELAY:       DURATION := -1.0 [s];
               in VERIFICATION_TIMEOUT:   DURATION := -1.0 [s];
               in WAIT_TIL_END:           BOOLEAN := FALSE;
               out SAS_ACK_CODE:          INTEGER;
               out SAS_ACK_TIME:          TIME;
               out VERIFICATION_STATUS:   CMD_VERIFICATION_STATUS;
               out CCSDS_SEQUENCE_COUNT:  INTEGER;
               out STATUS:                UCL_RETURN);

 -- Description:
 --   Issues a stimulus or Telecommand/Binary packet and verifies expected
 --   values for measurements, variables or derived values, wich are defined
 --   as part of the enditem definition in the MDB.
 --
 --   The verification starts after the command has been acknowledged
 --   and a specified time (ACTIVATION_DELAY) has elapsed. If in the
 --   VERIFICATION_TIMEOUT period all specified expected values have been
 --   verified, the verification status is set to OK, otherwise to FAILED.
 --   A message is generated accordingly.
 --
 --    Send   Activation delay  Verification timeout                        t
 --     -|---|----------------|-----------------------|---------------------->
 --        |   Acknowledge          |                   |
 --        |                        |                   |
 --         -> NACK                  -> all values OK    -> min one value NOK
 --
 --    Verification
 --    status: UNDEFINED               OK                  FAILED
 --
 --
 --   If during the verification an error occurs, the  verification status is
 --   set to ERROR.
 --
 --   If WAIT_TIL_END is set, the procedure waits until the end of the
 --   verification and the verification status is returned in
 --   VERIFICATION_STATUS; otherwise the status can be read by the procedure
 --   GET_VERIFICATION_STATUS at any time.
 -- Parameters:
 --   IN:
 --     ITEM                   : see above (procedure ISSUE)
 --     PRIO                   : see above (procedure ISSUE)
 --     AT_TIME                : see above (procedure ISSUE)
 --     BASE                   : see above (procedure ISSUE)
 --     ONBOARD_EXECUTION_TIME : see above (procedure ISSUE)
 --     TIMEOUT                : see above (procedure ISSUE)
 --     ACTIVATION_DELAY:  the time to wait after acknowledge of the GDU, until
 --                    the verification of the measurements, variables or
 --                    derived values is started. If this value is set to 0.0,
 --                    the checking is  performed immediately after the command
 --                    has been acknowledged. If the default value (-1.0) is
 --                    given, the activation delay is taken from the MDB
 --                    definition.
 --     VERIFICATION_TIMEOUT:  the time to wait after the ACTIVATION_DELAY
 --                    has expired and before the verification of the
 --                    measurements is finished. If in this period the
 --                    verification  of the  value(s) for measurement(s)
 --                    defined in the MDB is successful, a  message is generated
 --                    and the verification status is set to OK. If
 --                    VERIFICATION_TIMEOUT has elapsed and the verification
 --                    of at least one measurment is not successful, a  message
 --                    is generated indicating failure and the verification
 --                    status is set to FAILED.
 --                    If VERIFICATION_TIMEOUT is set to 0.0, the checking is
 --                    performed immediately after the ACTIVATION_DELAY and
 --                    then is finished.If the default value (-1.0) is given,
 --                    the activation delay is taken from the MDB definition.
 --     WAIT_TIL_END: If true, the procedure returns after the verification
 --                    has finished. If false, the procedure returns
 --                    immediately after the GDU has been acknowledged resp.
 --                    after TIMEOUT. In this case, the VERIFICATION_STATUS is
 --                    set to IN_PROGRESS.
 --
 --   OUT:
 --     SAS_ACK_CODE         : see above (procedure ISSUE)
 --     SAS_ACK_TIME         : see above (procedure ISSUE)
 --     VERIFICATION_STATUS  : Status of the verification
 --       STARTED     : Verification is started, i.e. Activation Delay is
 --                     currently running
 --       IN_PROGRESS : Verification is in progress, i.e Verification Timeout
 --                     is running and not all measurements have expected
 --                     values yet
```

Copyright per DIN 34

```
--       TC_VERIF_OK : Verification performed with success. All measurements
--                     had expected values.
--       TC_VERIF_FAILED : Verification performed with no success, i.e.
--                     Verification Timeout elapsed and at least one
--                     measurement had no expected value
--       ERROR       : Error during Verification Timeout. During the
--                     verification timeout an internal error occured
--       NO_VERIFICATION : No Verification defined for this enditem
--                     (i.e. in MDB there are no measurements, variables or
--                     derived values specified which to verify for this
--                     command)
--       ABORTED     : Verification aborted due to issue of same command or
--                     stop of remote TES
--       UNDEFINED   : The verifcation status is not defined, i.e. the sending
--                     of the command aborted due to error or negative
--                     acknowledge or timeout
--     CCSDS_SEQUENCE_COUNT : see above (procedure ISSUE)
--     STATUS               : see above (procedure ISSUE)

30:  procedure GET_VERIFICATION_STATUS
      (in ITEM:                 SINGLE_GDU_NAME;
       out VERIFICATION_STATUS: CMD_VERIFICATION_STATUS;
       out STATUS:              UCL_RETURN);

 -- Description:
 --   Read the verification status of an telecommand
 -- Parameters:
 --   IN:
 --     ENDITEM : name of stimulus or telecommand for which the verification
 --               status is to be fetched.
 --   OUT:
 --     VERIFICATION_STATUS  : see above (procedure ISSUE_AND_VERIFY)
 --     STATUS : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  INVALID_ITEM_NAME

31:  guarded procedure ENABLE_ENDITEM
      (in  ENDITEM: ISSUE_ITEM_LIST;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Enables an enditem that has been disabled, either temporarily through
 --   DISABLE_ENDITEM (see below), or permanently through definition in the
 --   MDB.
 -- Parameters:
 --   IN:
 --     ENDITEM : individual enditem or group of enditems to be enabled.
 --
 --         ENDITEM                | enable TC for sending for
 --         -----------------------+-----------------------------------------
 --         EGSE_ANALOG_STIMULUS   | single telecommand
 --         EGSE_DISCRETE_STIMULUS | single telecommand
 --         EGSE_BINARY_PACKET     | single telecommand
 --         PUS_TC                 | single telecommand
 --         EGSE_BINARY_PACKET     | single telecommand
 --         EGSE_PREDEFINED_TC     | single telecommand
 --         GDU_DESCRIPTION_LIST   | all telecommands from list
 --         VIRTUAL                | all telecommands under virtual node
 --         CDU                    | all telecommands under CDU node
 --   OUT:
 --     STATUS  : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  INVALID_ITEM_NAME

32:  guarded procedure DISABLE_ENDITEM
      (in  ENDITEM: ISSUE_ITEM_LIST;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Disables an enditem that has been enabled temporarily or permanently.
 -- Parameters:
 --   IN:
 --     ENDITEM : individual enditem or group of enditems to be disabled.
 --               see above (ENABLE_ENDITEM)
 --   OUT:
 --     STATUS  : UCL return status
 --                  OK
 --                  INVALID_TESTNODE_MODE
 --                  INVALID_ITEM_NAME



33:  guarded procedure SET_PRE_VALIDATION_CHECK
      (in  ENDITEM: TC_ITEM_LIST;
       in  SWITCH:  ON_OFF;
       out STATUS:  UCL_RETURN);


 -- Description:
```

Copyright per DIN 34

```
--   Enable/disables the pre validation check for an enditem / a group of
--   enditems.
--
--   Note: This procedure can deactivate the MDB configuration settings.
--
-- Parameters:
--   IN:
--     ENDITEM : individual enditem or group of enditems to be enabled.
--
--         ENDITEM                | enable TC for sending for
--         -----------------------+-----------------------------------------
--         EGSE_ANALOG_STIMULUS   | single telecommand
--         EGSE_DISCRETE_STIMULUS | single telecommand
--         EGSE_BINARY_PACKET     | single telecommand
--         PUS_TC                 | single telecommand
--         EGSE_BINARY_PACKET     | single telecommand
--         EGSE_PREDEFINED_TC     | single telecommand
--         SWOP_COMMAND           | single telecommand
--         GDU_DESCRIPTION_LIST   | all telecommands from list
--         VIRTUAL                | all telecommands under virtual node
--         CDU                    | all telecommands under CDU node
--
--     SWITCH  : switch to either ON or OFF
--   OUT:
--     STATUS  : UCL return status
--                   OK
--                   INVALID_PARAMETER      - enditem without definition
--                   INVALID_ITEM_NAME      - no such enditem of correct type
--                   INVALID_TESTNODE_MODE  - TES is not in rigth mode


34:  function PRE_VALIDATION_CHECK_ENABLED
      (ENDITEM: SINGLE_TC_NAME) : BOOLEAN;

 -- Description:
 --   Check if the pre validation check is enabled for an enditem.
 -- Parameters:
 --   OUT:
 --     Returns
 --       TRUE   if check for ENDITEM is enabled.
 --       FALSE  otherwise


35:  guarded procedure SET_TC_VERIFICATION_CHECK
      (in  ENDITEM: TC_ITEM_LIST;
       in  SWITCH:  ON_OFF;
       out STATUS:  UCL_RETURN);


 -- Description:
 --   Enable/disables the TC verification check for an enditem / a group of
 --   enditems.
 --
 --   Note: This procedure can deactivate the MDB configuration settings.
 --
 -- Parameters:
 --   IN:
 --     ENDITEM : individual enditem or group of enditems to be set.
 --               see above (SET_PRE_VALIDATION_CHECK)
 --     SWITCH  : switch to either ON or OFF
 --   OUT:
 --     STATUS  : UCL return status
 --                   INVALID_PARAMETER      - enditem without definition
 --                   INVALID_ITEM_NAME      - no such enditem of correct type
 --                   INVALID_TESTNODE_MODE  - TES is not in rigth mode


36:  function TC_VERIFICATION_CHECK_ENABLED
      (ENDITEM: SINGLE_TC_NAME) : BOOLEAN;

 -- Description:
 --   Check if the TC verification check is enabled for an enditem.
 -- Parameters:
 --   OUT:
 --     Returns
 --       TRUE   if check for ENDITEM is enabled.
 --       FALSE  otherwise


37:  guarded procedure SET_AUTHORIZATION_CHECK
      (guarded in  ENDITEM: SINGLE_TC_NAME;
               in  SWITCH:  ON_OFF;
               out STATUS:  UCL_RETURN);


 -- Description:
 --   Enable/disables the authorization check for critical commands for a
 --   single enditem.
 --
 --   Note: This procedure can deactivate the MDB configuration settings.
 --
```

Copyright per DIN 34

```
-- Parameters:
--   IN:
--     ENDITEM : individual enditem to enable/disable the authorization check.
--     SWITCH  : switch to either ON or OFF
--   OUT:
--     STATUS  : UCL return status
--                   INVALID_PARAMETER      - enditem without definition
--                   INVALID_ITEM_NAME      - no such enditem of correct type
--                   INVALID_TESTNODE_MODE  - TES is not in rigth mode

38:  function AUTHORIZATION_CHECK_ENABLED
      (ENDITEM: SINGLE_TC_NAME) : BOOLEAN;

-- Description:
--   Check if the authorization check is enabled for an enditem.
-- Parameters:
--   OUT:
--     Returns
--       TRUE   if check for ENDITEM is enabled.
--       FALSE  otherwise


39:  guarded procedure DOWNLOAD
      (in  APPL:   APPLICATION_ID;
       in  ITEM:   DOWNLOAD_ITEM_NAME;
       in  NODE:   NODE_NAME := \\;
       out STATUS: UCL_RETURN);

-- Description:
--   Transmit the name of a file to an application (SAS) to download it.
-- Parameters:
--   IN:
--     APPL : application ID (as returned from LOAD_APPLICATION) of the SAS
--            which is doing the download
--     ITEM : name of the MDB item to be downloaded, which refer to the file
--            to load
--     NODE : name of the node from which the item has to be downloaded,
--            by default the current node
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   ITEM_NOT_SAS_COMPATIBLE

-------------------------------------------------------------------------
-- Raw Data Handling
-------------------------------------------------------------------------

40:  guarded procedure START_ACQUISITION
      (in  ITEM:   ACQUISITION_COLLECTION;
       in  ADU:    ADU_NAME := \\;
       out STATUS: UCL_RETURN);

-- Description:
--   Starts data acquisition on the local test node for the given enditem(s).
--   This procedure works on measurement level.
-- Parameters:
--   IN:
--     ITEM : individual enditem or group of enditems to start acquisition.
--
--          ITEM               | stop acqusition for
--          -------------------+-------------------------------------------
--          MEASUREMENT        | single measurement
--          EGSE_MONITOR_LIST  | all measurements from list
--          PUS_STRUCTURE_ID   | all measurements from list
--          ADU_DESCRIPTION    | all measurements from ADU
--          VIRTUAL            | all measurements under virtual node
--          CDU                | all measurements under CDU node
--
--     ADU  : the ADU Descriptor to be applied
--            If there is only one ADU Descriptor defined for ITEM, the
--            parameter may be omitted.
--            If more than one ADU Descriptor is defined and the parameter is
--            omitted, the command starts acquisition for all ADUs.
--            If ITEM is already of type ADU_DESCRIPTION, the parameter is
--            ignored.
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   COMMUNICATION_ERROR
--                   INVALID_TESTNODE_MODE
--                   INVALID_ITEM_NAME
--                   NO_ADU_SERVICE
--                   INVALID_APPLICATION_NAME
--                   APPLICATION_NACK
--                   TIMEOUT
--                   APPLICATION_NOT_READY

41:  guarded procedure STOP_ACQUISITION
      (in  ITEM:   ACQUISITION_COLLECTION;
       in  ADU:    ADU_NAME := \\;
       out STATUS: UCL_RETURN);
```

Copyright per DIN 34

```
-- Description:
--   Stops data acquisition on the local test node for the given enditem(s).
--   This procedure works on measurement level. This procedure fails for
--   enditems, which are enabled for monitoring.
--
-- Remark:
--   For extension of this call see procedure FORCE_STOP_ACQUISITION.
--
-- Parameters:
--   IN:
--     ITEM : individual enditem or group of enditems to stop acquisition.
--
--         ITEM               | stop acqusition for
--         -------------------+--------------------------------------------
--         MEASUREMENT        | single measurement
--         EGSE_MONITOR_LIST  | all measurements from list
--         PUS_STRUCTURE_ID   | all measurements from list
--         ADU_DESCRIPTION    | all measurements from ADU
--         VIRTUAL            | all measurements under virtual node
--         CDU                | all measurements under CDU node
--
--     ADU  : the ADU Descriptor to be applied
--            If there is only one ADU Descriptor defined for ITEM, the
--            parameter may be omitted.
--            If more than one ADU Descriptor is defined and the parameter is
--            omitted, the command starts acquisition for all ADUs.
--            If ITEM is already of type ADU_DESCRIPTION, the parameter is
--            ignored.
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   ITEM_IS_ENABLED    (for monitoring)
--                   INVALID_TESTNODE_MODE
--                   INVALID_ITEM_NAME
--                   INVALID_APPLICATION_NAME
--                   APPLICATION_NACK
--                   TIMEOUT
--                   APPLICATION_NOT_READY


42:  function GET_ADU_STATUS (in ADU : ADU_NAME) : ADU_STATUS;


 -- Description:
 --   Get the adu status on the local test node for the given ADU.
 -- Parameters:
 --   IN:
 --     ADU  : name of ADU enditem
 --   OUT:
 --     ADU_STATUS (see description in GROUND_COMMON.ADU_STATUS)

43:  guarded procedure SEND_SIMULATED_ADU
      (in  ADU:    ADU_NAME;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Sends a specific ADU internally to the TES input buffers in simulation
 --   mode.
 -- Restriction:
 --   TES simulation mode only
 -- Parameters:
 --   IN:
 --     ADU    : pathname of ADU to be sent in simulation mode
 --              This ADU should be refered by a SIMULATED_ADU_DESCRIPTION.
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_ITEM_NAME
 --                   INVALID_ADU_TYPE

44:  procedure WAIT_FOR_ADU
      (in  ADU:             ADU_NAME;
       in  TIMEOUT:         DURATION := DEFAULT_ADU_WAIT_TIMEOUT;
       out TIME_TAG:        TIME;
       out SEQUENCE_NUMBER: INTEGER;
       out STATUS:          UCL_RETURN);

 -- Description:
 --   Waits for the reception of an ADU on the current test node.
 -- Parameters:
 --   IN:
 --     ADU     : name of packet to be received
 --     TIMEOUT : the maximum time to wait for reception of the ADU
 --               DEFAULT_ADU_WAIT_TIMEOUT ( = 0.0 [s])means no timeout, i.e.
 --               procedure waits up to the receiving of ADU or forever
 --   OUT:
 --     TIME_TAG        : the time tag from the received packet
 --     SEQUENCE_NUMBER : the current sequence number from the received packet
```

Copyright per DIN 34

```
--     STATUS          : UCL return status
--                   OK
--                   INVALID_TESTNODE_MODE
--                   INVALID_AP_ID                    ??????????
--                   INVALID_ITEM_NAME
--                   ITEMS_NOT_ACQUIRED
--                   TIMEOUT
--                   OPERATION_ABORTED
--                   INVALID_ADU_TYPE

45:  guarded procedure SET_BITS_IN_SIMULATED_ADU
      (in  ADU:    ADU_NAME;
       in  OFFSET: INTEGER;
       in  SIZE:   BIT_COUNT;
       in  VALUE:  BITSET;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Modifies the bit-contents of a simulated ADU directly in simulation mode.
 -- Restriction:
 --   The operation can only be applied on unstructured ADUs or CCSDS ADUs.
 --   TES simulation mode only
 -- Parameters:
 --   IN:
 --     ADU    : pathname of ADU to be modified in simulation mode
 --              This ADU should be refered by a SIMULATED_ADU_DESCRIPTION.
 --     OFFSET : location in bits in this ADU, where bits are to be modified
 --     SIZE   : the number of bits to modify. The upper limit is 32 bits
 --     VALUE  : the new value of the set of bits to be modified
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_ITEM_NAME
 --                   INVALID_ADU_TYPE

46:  guarded procedure SET_SIMULATED_ENDITEM_VALUE
      (in  ITEM:            MONITOR_ITEM_NAME;
       in  BOOL_VALUE:      BOOLEAN :=  FALSE;
       in  INT_VALUE:       INTEGER :=  0;
       in  BITSET_VALUE:    BITSET :=  {};
       in  REAL_VALUE:      REAL :=  0.0;
       in  LONG_REAL_VALUE: LONG_REAL :=  0.0;
       in  STR_VALUE:       USER_MESSAGE :=  "";
       out STATUS:          UCL_RETURN);

 -- Description:
 --   Modifies the value of an enditem in simulated ADUs.
 -- Restriction:
 --   The operation can only be applied on measurements provided by structured
 --   ADUs.
 --   TES simulation mode only
 -- Parameters:
 --   IN:
 --     ITEM : pathname of item to be modified.
 --     Other parameters : To cope with the different data types a value can
 --            have, several parameters are provided. By selecting the
 --            appropriate parameter, eg INT_VALUE: 7, the desired value is
 --            set. The operation can only be applied on items provided by
 --            structured ADUs.
 --      Note: to set simulated enditem values of type UNSIGNED_INTEGER, the
 --            parameter BITSET_VALUE has to be used. A BITSET_VALUE can be
 --            achieved by using type conversion:
 --            BITSET_VALUE: BITSET(U_INT_VALUE)
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_ITEM_NAME
 --                   INVALID_ADU_TYPE

47:  guarded procedure ROUTE_TO_SAS
      (in  ITEM:         ROUTE_SAS_ITEM_NAME ;
       in  SAS_NAME:     APPLICATION_NAME;
       in  OLD_SAS_NAME: APPLICATION_NAME :=  NULL_APPLICATION_NAME;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   Routes all items specified in ITEM which have an SAS Reference to the
 --   new SAS specified by SAS_NAME.
 -- Restriction:
 --   Items of type ADU_DESCRIPTION must be not acquired for this operation.
 -- Parameters:
 --   IN:
 --     ITEM : The item to be changed on the test node
 --
 --         ITEM                 | change SAS reference for
 --         ---------------------+-----------------------------------------
 --         ADU_DESCRIPTION      | single ADU
 --         GDU (telecommand)    | single GDU
 --         GDU_DESCRIPTION_LIST | all GDU from list
```

Copyright per DIN 34

```
--         VIRTUAL              | all ADU / GDU under virtual node
--         CDU                  | all ADU / GDU under CDU node
--
--     SAS_NAME     : The new short name of the SAS to be applied for ITEM
--     OLD_SAS_NAME : The old short name  of the SAS. Only items specified
--                    in ITEM having a SAS Reference matching this SAS name
--                    are changed.
--                    If NULL_APPLICATION_NAME is given, all items specified
--                    in ITEM are changed.
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   NOT_OK    (not all from set are changed)
--                   INVALID_APPLICATION_NAME
--                   INVALID_TESTNODE_MODE
--                   INVALID_ITEM_NAME

48:   guarded procedure SWITCH_SAS_REFERENCE
      (in  ITEM:         ROUTE_SAS_ITEM_NAME ;
       in  SAS_NAME:     APPLICATION_NAME;
       in  OLD_SAS_NAME: APPLICATION_NAME :=  NULL_APPLICATION_NAME;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   Reroutes all items specified in ITEM having a SAS Reference to the
 --   new SAS specified by SAS_NAME. For ADU and VIRTUAL nodes containing ADU
 --   a STOP_ACQUISITION will be automatically sent to the old SAS. After
 --   having rerouted the items to the new SAS, a START_ACQUISITION will be
 --   automatically sent to that new SAS. Note: the monitoring status is not
 --   changed by TES, i.e. the limit sets for the monitored items will still
 --   be the same.
 --
 -- Remark:
 --   To acquire all Telemetry Data (ADUs) from a specific SAS initially you
 --   can call simple:  SWITCH_SAS_REFERENCE (\,"SAS","SAS",Status);
 --   where SAS is the name of the specific SAS.
 --
 -- Remark (1):
 --   To avoid unwanted acquisition of not acquired Telemetry Data (ADUs) you
 --   can call: ROUTE_TO_SAS (\,"NOT_EXISTING_SAS","OLD_SAS",Status);
 --             SWITCH_SAS_REFERENCE (\,"NEW_SAS","OLD_SAS", Status);
 --             ROUTE_TO_SAS (\,"NOT_EXISTING_SAS","NEW_SAS",Status);
 --
 -- Restriction:
 --   Items of type TC and VIRTUAL, CDU containing TCs: same behaviour as
 --   procedure ROUTE_TO_SAS
 -- Parameters:
 --   IN:
 --     ITEM : The item to be changed on the test node
 --
 --         ITEM                 | change SAS reference for
 --         ---------------------+-------------------------------------------
 --         ADU_DESCRIPTION      | single ADU
 --         GDU (telecommand)    | single GDU
 --         GDU_DESCRIPTION_LIST | all GDU from list
 --         VIRTUAL              | all ADU / GDU under virtual node
 --         CDU                  | all ADU / GDU under CDU node
 --
 --     SAS_NAME     : The new short name of the SAS to be applied for ITEM
 --     OLD_SAS_NAME : The old short name  of the SAS. Only items specified
 --                    in ITEM having a SAS Reference matching this SAS name
 --                    are changed.
 --                    If NULL_APPLICATION_NAME is given, all items specified
 --                    in ITEM are changed.
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   NOT_OK    (not all from set are changed)
 --                   INVALID_APPLICATION_NAME
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_ITEM_NAME
 --                   ITEM_UNKNOWN
 --                   NO_ADU_SERVICE
 --
 --    In case the acquisition of a single item cannot be stopped the return
 --    status as defined for procedure STOP_ACQUISITION will be returned.
 --    If the acquisition of a single item cannot be started on the new SAS, the
 --    return status as defined by procedure START_ACQUISITION will be returned.
 --    In that case the specified item will be assigned to the new SAS and the
 --    monitoring may still be active, but no acquisition will be performed.

49:   procedure GET_SAS_REFERENCE
      (in  ITEM:     ROUTE_SAS_ITEM_NAME;
       out SAS_NAME: APPLICATION_NAME;
       out STATUS:   UCL_RETURN);

 -- Description:
 --   Returns the name of the SAS to which the specified item has been assigned.
 -- Parameters:
 --   IN:
 --     ITEM : The item, for which the SAS reference is to returned back.
```

Copyright per DIN 34

```
--   OUT:
--     SAS_NAME : The short name of the SAS
--     STATUS   : UCL return status
--                   OK
--                   INVALID_ITEM_NAME (no sas reference or unknown item)

50:  guarded procedure SET_CCSDS_APID
      (in  ITEM:           CCSDS_ITEM_NAME;
       in  CCSDS_APID:     CCSDS_APID_RANGE;
       in  OLD_CCSDS_APID: OLD_APID_RANGE := -1;
       out STATUS:         UCL_RETURN);

-- Description:
--   Change the application ID (APID) for CCSDS packets.
-- Restriction:
--   Items of type ADU_DESCRIPTION must be not acquired for this operation.
-- Parameters:
--   IN:
--     ITEM : The item to be changed on the test node
--
--          ITEM                 | change SAS reference for
--          ---------------------+-----------------------------------------
--          ADU_DESCRIPTION      | single ADU
--          GDU (telecommand)    | single GDU
--          GDU_DESCRIPTION_LIST | all GDU from list
--          VIRTUAL              | all ADU / GDU under virtual node
--          CDU                  | all ADU / GDU under CDU node
--
--     CCSDS_APID    : The new APID to be applied for the CCSDS Primary
--                     Header contained in ITEM.
--     OLD_CCSDS_APID: The old APID  of the item.Only items specified in ITEM
--                     having an APID matching this OLD_CCSDS_APID are
--                     changed .
--                     If a negative value is given, all items  specified in
--                     ITEM are changed.
--       Note: The range for parameter CCSDS_APID is 0..4095 to cover extended
--             APID: for values greater than 2047 the CCSDS Packet type bit
--             (extension-bit) will be set.
--             This behaviour can be influenced by the CGS configuration
--             parameter TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID: the
--             CCSDS Packet type bit will not be set if the value of this
--             parameter is set to FALSE. The default value of
--             TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID is TRUE.
--   OUT:
--     STATUS : UCL return status
--                   OK
--                   NOT_OK    (not all from set are changed)
--                   INVALID_TESTNODE_MODE
--                   INVALID_ITEM_NAME


-------------------------------------------------------------------------
-- Log Event Handling
-------------------------------------------------------------------------

51:  procedure LOG (in  MESSAGE:   USER_MESSAGE;
                    in  LONG_TEXT: USER_MESSAGE := "";
                    out STATUS:    UCL_RETURN);

-- Description:
--   Logs an event message to the test result database.
--   Logged events have group-code LOG and type-code MSG.
-- Parameters:
--   IN:
--     MESSAGE   : string of up to 255 characters containing the message to
--                 be logged. First 40 characters of string will appear in
--                 the Short_Text field of logged event.
--        Note: If message is longer than 40 characters, and LONG_TEXT is
--              empty, whole MESSAGE will be duplicated to appear as well
--              in long text field of logged event .
--     LONG_TEXT : Additional message to be sent to user. Appears in the
--                 Long_Text field of the logged event.
--   OUT:
--     STATUS : UCL return status
--                   OK

52:  procedure USER_EVENT
      (in  MESSAGE: USER_MESSAGE;
       out STATUS:  UCL_RETURN);

-- Description:
--   Logs an user event message to the test result database.
--   User events are special log event with the group-code UEVT and
--   type-code MSG. They can be used later on test evaluation to select
--   timeframes.
-- Parameters:
--   IN:
--     MESSAGE   : string of up to 255 characters containing the message to
--                 be logged. First 40 characters of string will appear in
--                 the Short_Text field of logged user event.
--                 The whole MESSAGE will be duplicated to appear as well
```

Copyright per DIN 34

```
--                in long text field of logged user event.
--   OUT:
--     STATUS : UCL return status
--                   OK


-------------------------------------------------------------------------
-- Archiving
-------------------------------------------------------------------------


53:  procedure ENABLE_ARCHIVING
      (in  CYCLE:  DURATION := DEFAULT_ARCH_FILE_OPEN_PERIOD;
       in  KIND:   ARCHIVE_COLLECTION := ALL_PACKETS;
       in  ITEM:   ARCHIVE_ITEM_NAME := \\;
       out STATUS: UCL_RETURN);


 -- Description:
 --   Enables archiving for all packets of KIND on the local test node and
 --   creates a new archive file.
 -- Parameters:
 --   IN:
 --     CYCLE : the time interval (1 minute to 24 hours exclusive) at which
 --             archive files are closed automatically. If archiving is already
 --             enabled and the cycle time of this call differs from the cycle
 --             time specified in the last call of this procedure, the current
 --             archive file is closed, and a new one is opened. Archiving
 --             periodically closes the archive file using the new cycle time.
 --     KIND  : kind of packets, either TM_PACKETS, TC_PACKETS or both
 --             Each call add the selected ARCHIVE_COLLECTION  to the previous
 --             settings. To remove special ARCHIVE_COLLECTION use call
 --             DISABLE_ARCHIVING.
 --     ITEM  : PATHNAME of ITEM, that should be archived. Possible:
 --             ADU, GDU, SWOP, RESPONSE, virtual
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_CYCLE
 --                   RUNTIME_ERROR


54:  procedure DISABLE_ARCHIVING
      (in  KIND: ARCHIVE_COLLECTION := ALL_PACKETS;
       in  ITEM: ARCHIVE_ITEM_NAME := \\);


 -- Description:
 --   Stops archiving for all packets of KIND on the local test node. If
 --   archiving was enabled the current archive file is closed.
 -- Parameters:
 --   IN:
 --     KIND  : kind of packets, either TM_PACKETS, TC_PACKETS or both
 --             Each call remove the selected ARCHIVE_COLLECTION  from the previous
 --             settings. To disable archiving at all, use KIND = ALL_PACKETS.
 --     ITEM  : PATHNAME of ITEM, that should not be archived anymore. Possible:
 --             ADU, GDU, SWOP, RESPONSE, virtual

55:  procedure CLOSE_ARCHIVE
      (out CLOSE_TIME: TIME;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Close the current open archive file and open a new one on the local test
 --   node.
 -- Parameters:
 --   OUT:
 --     CLOSE_TIME : time of closing the archive file
 --     STATUS     : UCL return status
 --                   OK
 --                   ARCHIVING_IS_DISABLED
 --                   RUNTIME_ERROR

-------------------------------------------------------------------------
-- TRDB Access
-------------------------------------------------------------------------

56:  procedure ADD_USER_FILE_TO_TEST_SESSION
      (in  FILE_NAME:     STRING;
       in  FILE_TYPE:     STRING := "ANY     ";
       in  SESSION:       STRING := "";
       in  PRODUCER:      STRING := "";
       in  CREATION_TIME: TIME   := ~:~;
       out SESSION_NAME:  T_SESSION_NAME;
       out STATUS:        UCL_RETURN);

 -- Description:
 --   Add a file to TRDB in current open session.
 --   The file denoted by FILE_NAME is copied into the TRDB, the name of the
 --   currently active session is returned.
 -- Note: the copy operation into the TRDB filesystem is decoupled from the
```

Copyright per DIN 34

```
--       checks done to verify e.g. that the file has not already been transferred
--       into the TRDB, or whether the file is accessible on the DB server,
--       in order not to block the calling application.
--       This means a status OK might be returned although the file has not yet
--       been copied into the TRDB. The following operation RETRIEVE_FILENAME_FROM_TRDB
--       can be used to verify this.
-- Parameters:
--   IN:
--     FILE_NAME: the pathname of the file to be copied, must be an absolute
--                path and be visible on the DB server node (environment
--                variable can be resolved)
--     FILE_TYPE: a string of 8 characters which is assigned to the file in
--                the TRDB, may be used as selection criteria
--     SESSION:   the session the file will be added to.
--                default = "" means the session matching to the given time.
--     PRODUCER:  a string for the producer
--                default = "" means current application (ie. TES_xx).
--     CREATION_TIME:
--                the pathname of the file to be copied, must be an absolute
--                default = ~:~ means current time.
--   OUT:
--     SESSION_NAME : the name of the currently active session, the file is
--                    added to
--     STATUS       : UCL return status
--                    OK
--                    NOT_OK
--                    TIMEOUT
--                    INVALID_PARAMETER
--                    EXISTING_FILE
--                    FILE_NOT_FOUND

57:  procedure RETRIEVE_FILENAME_FROM_TRDB
      (in  SESSION_NAME: T_SESSION_NAME;
       in  FILE_NAME:    STRING;
       out LOCATION:     STRING;
       out STATUS:       UCL_RETURN);

-- Description:
--   Retrieve the file trom TRDB.
--   The complete pathname of the file as can be found in the TRDB will be
--   returned.
-- Parameters:
--   IN:
--     SESSION_NAME : name of the session, where the file can be found within
--                    the TRDB
--     FILE_NAME    : the name of the file, note: only the basename of the
--                    file has to be passed here
--   OUT:
--     LOCATION : the complete pathname of the file in the TRDB (including the
--                basename of the file)
--     STATUS   : UCL return status
--                    OK
--                    NOT_OK
--                    TIMEOUT
--                    INVALID_PARAMETER
--                    FILE_NOT_FOUND

-------------------------------------------------------------------------
-- User Input & Output
-------------------------------------------------------------------------

58:  procedure READ_MESSAGE_FROM_USER
      (in  PROMPT:      USER_MESSAGE;
       in  WORKSTATION: NODE_NAME := \\;
       in  OPTIONS:     VALUE_STRING := "";
       out USER_ENTRY:  USER_MESSAGE;
       out STATUS:      UCL_RETURN);

-- Description:
--   Reads a message from the user; issues a prompt and blocks until the
--   user enters a message.
-- Restriction:
--    UCL only – not usable from HLCL
-- Parameters:
--   IN:
--     PROMPT      : prompt string of up to 255 characters, on Workstation's
--                   screen.
--     WORKSTATION : node to display the prompt
--                   by default, the workstation that started this AP
--                   or
--                   if not started from workstation the default workstation
--                   (see library call SET_DEFAULT_WORKSTATION)
--     OPTIONS     : options specifying attributes for Pop–Up window
--
--             Syntax: –position X Y
--                     –size W H
--                     –foreground <colour>
--                     –image <name>
--
--                     X,Y:  coordinates; W: Width, H: Height
--                     <colour>: colour name or hexadecimal specification
```

Copyright per DIN 34

```
--                    <name>: image name, valid values are info, warning, error, question,
--                            or any Gtk stock item name link gtk-home, gtk-network, gtk-refresh.
--
--             Example: "-position 100 200 -size 600 150 -foreground red"
--
--   OUT:
--     USER_ENTRY : returns the user input (string)
--     STATUS     : UCL return status
--                    OK
--                    WORKSTATION_NOT_CONNECTED
--                    WORKSTATION_NOT_READY
--                    INPUT_CANCELLED
--                    INVALID_TESTNODE_MODE

59:  procedure WRITE_MESSAGE_TO_USER
      (in  MESSAGE:           USER_MESSAGE;
       in  SUPPLEMENT:        USER_MESSAGE := "";
       in  MESSAGE_GROUP:     MSG_GROUP := "";
       in  MESSAGE_TYPE:      MSG_TYPE := "MSG ";
       in  WORKSTATION:       NODE_NAME :=\\;
       in  ALL_WORKSTATIONS : BOOLEAN := FALSE;
       out STATUS:            UCL_RETURN);

 -- Description:
 --   Sends a message to the user of a given or all workstation(s).
 -- Restriction:
 --    UCL only - not usable from HLCL
 -- Parameters:
 --   IN:
 --     MESSAGE    : Message to be sent to user. Appears in the (short) text
 --                  field of the message handler window. Is abbreviated to
 --                  first 40 characters.
 --
 --       Note: If message is longer than 40 characters, and SUPPLEMENT is
 --             empty, whole MESSAGE will be duplicated to appear as well in
 --             supplement text field of message handler window .
 --
 --     SUPPLEMENT : Additional message to be sent to user. Appears in the
 --                  supplement text field of the message handler window.
 --     MSG_GROUP  : A string of 4 character defining the message group.
 --                  Might be one of the predefined CGS message groups, or
 --                  defined  by the user .
 --     MSG_TYPE   : A string of 4 character defining the message type.
 --                  Message Types are mapped to the message class of the
 --                  message handler.
 --
 --            The following are defined:
 --            "INFO":         mapped to ADVISORY
 --            "MSG ":         mapped to ADVISORY
 --            "ERR ":         mapped to ORDINARY
 --            "WRN ":         mapped to ADVISORY
 --            "EXC ":         mapped to SEVERE
 --                                (to be used only for monitoring exceptions)
 --            "ALRM":         mapped to FATAL
 --            any other:      mapped to ADVISORY
 --
 --     WORKSTATION : name of node to display message; by default the
 --                   workstation where this AP (or its ancestors) was started.
 --     ALL_WORKSTATIONS : if TRUE, MESSAGE is sent to all workstations
 --                   connected to the test node. Parameter WORKSTATION is
 --                   ignored in this case.
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   WORKSTATION_NOT_CONNECTED
 --                   WORKSTATION_NOT_READY
 --                   COMMUNICATION_ERROR
 --                   PROTOCOL_ERROR

60:  procedure READ_NUMBER_FROM_USER
      (in  PROMPT_MESSAGE: USER_MESSAGE;
       in  WORKSTATION:    NODE_NAME := \\;
       in  OPTIONS:        VALUE_STRING := "";
       out USER_ENTRY:     REAL;
       out STATUS:         UCL_RETURN);

 -- Description:
 --   Reads a real number; issues a prompt and blocks until the user enters
 --   one at the workstation.
 -- Restriction:
 --    UCL only - not usable from HLCL
 -- Parameters:
 --   IN:
 --     PROMPT      : see above (READ_MESSAGE_FROM_USER)
 --     WORKSTATION : see above (READ_MESSAGE_FROM_USER)
 --     OPTIONS     : see above (READ_MESSAGE_FROM_USER)
 --   OUT:
 --     USER_ENTRY : returns the user input (float)
 --     STATUS     : UCL return status
 --                    OK
 --                    WORKSTATION_NOT_CONNECTED
```

Copyright per DIN 34

```
--                  WORKSTATION_NOT_READY
--                  INPUT_CANCELLED
--                  INVALID_TESTNODE_MODE


-------------------------------------------------------------------------
-- Synoptic Display and Workstation Program Control
-------------------------------------------------------------------------

61:  procedure DISPLAY_PICTURE
      (in  PICTURE:     PICTURE_NAME;
       in  WORKSTATION: NODE_NAME := \\;
       in  OPTIONS:     VALUE_STRING := "";
       out ID:          PICTURE_ID;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Displays a synoptic display on a specific workstation.
 -- Parameters:
 --   IN:
 --     PICTURE     : name of synoptic display (pathname)
 --     WORKSTATION : the work station (pathname) to show the synoptic display:
 --                   by default the work station from which the command
 --                   originates, directly or via a AP.
 --     OPTIONS     : options specifying attributes for window where picture
 --                   is displayed
 --
 --             Syntax: –position X Y
 --                     –size W H
 --                     –docked | –undocked (only applicable for USS displays)
 --
 --                     X,Y:  coordinates; W: Width, H: Height
 --
 --             Example: "–position 100 200 –size 600 150 –undocked"
 --
 --   OUT:
 --     ID     : returns identifier to be used by REMOVE_PICTURE when removing
 --              this synoptic display.
 --     STATUS : UCL return status
 --                   OK
 --                   WORKSTATION_NOT_CONNECTED
 --                   WORKSTATION_NOT_READY
 --                   SYNOPTIC_RESOURCES_EXCEEDED
 --                   TIMEOUT

62:  procedure REMOVE_PICTURE
      (in  ID:          PICTURE_ID;
       in  WORKSTATION: NODE_NAME := \\;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Remove a synoptic display from the screen of a specific workstation.
 -- Parameters:
 --   IN:
 --     ID          : identifier, as returned from DISPLAY_PICTURE
 --     WORKSTATION : the work station showing the synoptic display: by default
 --                   the work station from which the command originates,
 --                   directly or via a AP.
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   WORKSTATION_NOT_CONNECTED
 --                   WORKSTATION_NOT_READY


63:  guarded procedure START_PROGRAM
      (in  WORKSTATION: NODE_NAME := \\;
       in  PROGRAM:     STRING;
       out PID:         INTEGER;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Starts a UNIX program (application) on a workstation (via HCI)
 --   It is further possible to start an HCI application (i.e. one of the
 --   windows within HCI).
 -- Parameters:
 --   IN:
 --     WORKSTATION : the work station starting the program: by default
 --                   the work station from which the command originates,
 --                   directly or via a AP
 --     PROGRAM     : the file name of the program plus (optional) start up
 --                   parameters according to UNIX conventions resp. acc. to
 --                   the program's startup profile
 --
 --       Note: For a list of allowed HCI application names refer the
 --             CGS User Manual. The application name has to be given with the
 --             prefix "HCI.".
 --             For the parameters that are allowed for  HCI applications
 --             refer also CGS User Manual.
 --
 --   OUT:
```

Copyright per DIN 34

```
--     PID    : the ID of the UNIX process / HCI window created
--     STATUS : UCL return status
--                  OK
--                  NOT_OK

64:  guarded procedure SIGNAL_PROGRAM
      (in  WORKSTATION: NODE_NAME := \\;
       in  PID:         INTEGER;
       in  SIGNAL:      STRING;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Sends a UNIX signal to a program (application) on a workstation (via HCI)
 --   or to an HCI window.
 -- Parameters:
 --   IN:
 --     WORKSTATION : the work station the program is running: by default
 --                   the work station from which the command originates,
 --                   directly or via a AP
 --     PID         : the ID of the process (as received in START_PROGRAM)
 --     SIGNAL      :  The name the signal(s) to be sent
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   NOT_OK

65:  procedure OPEN_AP_OUTPUT_WINDOW
      (in  WORKSTATION:       NODE_NAME := \\;
       in  WINDOW_PROPERTIES: STRING := "";
       out PID:               INTEGER;
       out STATUS:            UCL_RETURN);

 -- Description:
 --   Starts the message handler on a workstation (via HCI), which opens  the
 --   message window with a filter to select only messages generated by the AP
 --   calling this procedure.
 -- Restriction:
 --    UCL only – not usable from HLCL
 -- Parameters:
 --   IN:
 --     WORKSTATION : the work station starting the message handler: by default
 --                   the work station from which the command originates,
 --                   directly or via a AP
 --     WINDOW_PROPERTIES : The name of the predefined (saved) tool properties,
 --                   which the message  handler shall apply when opening the
 --                   message window.
 --                   In case the default ("") is given, the message handler
 --                   starts with standard properties
 --                   The filter to select the AP messages is overriding the
 --                   filter defined in the properties for the field "extra".
 --   OUT:
 --     PID    : the process ID of the message window
 --     STATUS : UCL return status
 --                   OK
 --                   NOT_OK

66:  procedure CLOSE_AP_OUTPUT_WINDOW
      (in  WORKSTATION: NODE_NAME := \\;
       in  PID:         INTEGER;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   Stops the message handler on a workstation (via HCI)
 -- Parameters:
 --   IN:
 --     WORKSTATION : the work station running the message handler: by default
 --                   the work station from which the command originates,
 --                   directly or via a AP
 --     PID         : the process ID of the message window (returned by
 --                   OPEN_AP_OUTPUT_WINDOW)
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   NOT_OK


 ------------------------------------------------------------------------
 -- Test Node management
 ------------------------------------------------------------------------

67:  function IS_LOCAL_NODE (in NODE : NODE_NAME) : BOOLEAN;

 -- Description:
 --   Returns TRUE if executed on the test node NODE, otherwise FALSE.
 -- Parameters:
 --   IN:
 --     NODE : name of EGSE node

68:  guarded procedure SET_DEFAULT_WORKSTATION
      (in  WORKSTATION: NODE_NAME :=\\;
       out STATUS:      UCL_RETURN);
```

Copyright per DIN 34

```
 -- Description:
 --  Overwrite the default workstation for a test node.
 --
 --  The default workstation is set initially by CGS configuration parameter
 --  TES.KERNEL.DEFAULT_WORKSTATION.
 --
 --  The default workstation is used by all APs that are not started
 --  via interactive commands (e.g. APs started via monitoring
 --  exceptions/conditions or API call).
 -- Parameters:
 --   IN:
 --     WORKSTATION : the work station forseen as default work station: by
 --                   default the work station from which the command
 --                   originates, directly or via a AP
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   WORKSTATION_NOT_CONNECTED
 --                   INVALID_ITEM_NAME

69:  function GET_NODE_FOR_ITEM  (ITEM: PATHNAME): NODE_NAME;

 -- Description:
 --   Returns the actual node which maintains the specified item.
 --
 --   Note:  If the specified item is not assigned or unknown to
 --          a testnode the function will return \\.
 --
 -- Parameters:
 --   IN:
 --     ITEM : pathname of enditem for query

70:  guarded procedure FORCE_STOP_ACQUISITION
      (in  ITEM:   ACQUISITION_COLLECTION;
       in  ADU:    ADU_NAME := \\;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Stops data acquisition on the local test node for the given enditem(s).
 --   This procedure works on measurement level. This procedure enforce stop
 --   acuisition, also for enditems, which are enabled for monitoring! The
 --   monitoring state of these enditems is unchanged.
 --
 -- Remark:
 --   For a checked procedure with respect to monitored values see procedure
 --   STOP_ACQUISITION.
 --
 -- Parameters:
 --   IN:
 --     ITEM : individual enditem or group of enditems to stop acquisition.
 --
 --         ITEM               | stop acqusition for
 --         -------------------+-------------------------------------------
 --         MEASUREMENT        | single measurement
 --         EGSE_MONITOR_LIST  | all measurements from list
 --         PUS_STRUCTURE_ID   | all measurements from list
 --         ADU_DESCRIPTION    | all measurements from ADU
 --         VIRTUAL            | all measurements under virtual node
 --         CDU                | all measurements under CDU node
 --
 --     ADU  : the ADU Descriptor to be applied
 --            If there is only one ADU Descriptor defined for ITEM, the
 --            parameter may be omitted.
 --            If more than one ADU Descriptor is defined and the parameter is
 --            omitted, the command starts acquisition for all ADUs.
 --            If ITEM is already of type ADU_DESCRIPTION, the parameter is
 --            ignored.
 --   OUT:
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   INVALID_ITEM_NAME
 --                   INVALID_APPLICATION_NAME
 --                   APPLICATION_NACK
 --                   TIMEOUT
 --                   APPLICATION_NOT_READY


END GROUND_LIBRARY;
```

Copyright per DIN 34

I-3 UCL System library: GRD_CONVERSION

The GRD_CONVERSION **library** contains all standard **procedures** used for conversion of items within the Ground Automated **procedures** / Sequences

I-3.1 UCL Ground System library Specification: GRD_CONVERSION

```
------------------------------------------------------------------------
--
-- ****************************************
--  GRD_CONVERSION UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to convert items within the GROUND system
--
--     Must be compiled for ground and with Body Id = 11
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : GRD_CONVERSION System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/grd_conversion_.ucl#5 $"
--*----------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library GRD_CONVERSION;

------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  1:  procedure PATHNAME_TO_STRING
        (in  PATH: PATHNAME;
         out NAME: STRING);

  -- Description:
  --   convert the pathname type into a string type
  -- Restriction:
  --   If PATH is not existing on testnode scope, an empty string is returned.
  --   If NAME is too short to hold resulting string, the string is truncated.

  2:  function PATH (in NAME: STRING) : PATHNAME;

  -- Description:
  --   convert the string type into a pathname type
  -- Restriction:
  --   returns Null-Pathname in case the path is not existing on testnode scope


  3:  procedure STATECODE_TO_STRING
        (in  CODE:      STATECODE;
         out CODE_TEXT: STRING);

  -- Description:
  --   convert the statecode type into a string type
  -- Restriction:
  --   If CODE_TEXT is too short to hold resulting string, the string is
  --   truncated.
  --   If CODE_TEXT is longer than 8 characters, it is filled with blanks.

  4:  function CODE (in CODE_TEXT : STRING) : STATECODE;

  -- Description:
  --   convert the string type into a statecode type
  -- Restriction:
  --   If CODE_TEXT is longer than 8 characters, the string is truncated.
  --   If CODE_TEXT is shorter than 8 characters, the STATECODE is extended by
  --   blanks.


END GRD_CONVERSION;
```

Copyright per DIN 34

I-4 UCL Ground Commands To Onboard System library

This **library** contains commands sent to the COF DMS system.

library Id (Body_Id): 4

I-4.1 UCL GROUND_TO_OB_LIB System library Specification

```
--
--*****************************************
-- GROUND_TO_OB_LIB System Library
--*****************************************
--ABSTRACT--
--     Defines Procedures to send commands to the onboard system
--     Must be compiled with body ID = 4
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : GROUND_TO_OB_LIB System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/ground_to_ob_lib_.ucl#5 $"
--*-------------------------------------------------------------------
--CONTENTS--
--      COMPILER     : UCLC
--      LANGUAGE     : UCL
--
-------------------------------------------------------------------------

library GROUND_TO_OB_LIB;

-------------------------------------------------------------------------
-- IMPORTS
-------------------------------------------------------------------------

  import GROUND_COMMON;

-------------------------------------------------------------------------
-- CONSTANTS AND TYPES
-------------------------------------------------------------------------

  type INSTANCE   = PATHNAME (CCSDS_END_POINT);
  type END_POINT  = PATHNAME (CCSDS_END_POINT);
  type FLAP_NAME  = PATHNAME (UCL_AUTOMATED_PROCEDURE);
  type SW_COMMAND = PATHNAME (SWOP_COMMAND);

  type SW_COMMANDS_AND_RESPONSES = PATHNAME (SWOP_COMMAND,
                                             RESPONSE_PACKET,
                                             VIRTUAL);


  --
  -- Type and Constants defining priority
  --

  type PRIORITY = (LOW, HIGH);

  constant ONBOARD_LOW:                   INTEGER := 1;
  constant ONBOARD_HIGH:                  INTEGER := 2;
  constant DEFAULT_ONBOARD_PRIORITY:      INTEGER := 0;

  type ONBOARD_PRIORITY = INTEGER (DEFAULT_ONBOARD_PRIORITY .. ONBOARD_HIGH);

  --
  -- Constants defining timeouts
  --

  constant DEFAULT_SW_COMMAND_TIMEOUT: DURATION := 10.0 [s];
  constant DEFAULT_FLAP_TIMEOUT:       DURATION := 30.0 [s];

  --
  -- Types and Constants defining the status returned in the response packet
  --

  type FLAP_RETURN = INTEGER;
  type ONBOARD_RETURN = INTEGER;
  type FLAP_ID  = PATHNAME (EGSE_INTEGER_MEASUREMENT);

  constant ONB_RESULT_SUCCESS:                  ONBOARD_RETURN :=  0;
  constant ONB_RESULT_SEQUENCE_COUNT_ANOMALY:   ONBOARD_RETURN :=  1;
  constant ONB_RESULT_NOT_AUTHORIZED:           ONBOARD_RETURN :=  2;
  constant ONB_RESULT_UNKNOWN_COMMAND:          ONBOARD_RETURN :=  3;
  constant ONB_RESULT_COMMAND_NOT_VALID:        ONBOARD_RETURN :=  4;
  constant ONB_RESULT_PARAMETERS_INCOMPLETE:    ONBOARD_RETURN :=  5;
  constant ONB_RESULT_PARAMETERS_NOT_VALID:     ONBOARD_RETURN :=  6;
  constant ONB_RESULT_SEQUENCE_COUNT_REPETITION: ONBOARD_RETURN :=  7;
```

Copyright per DIN 34

```
constant ONB_RESULT_SEQUENCE_COUNT_MISMATCH:   ONBOARD_RETURN :=  8;
constant ONB_RESULT_CHECKSUM_ERROR:            ONBOARD_RETURN :=  9;
constant ONB_RESULT_TWO_STAGE_BUF_FULL:        ONBOARD_RETURN :=  10;
constant ONB_RESULT_TWO_STAGE_SOURCE_MISMATCH: ONBOARD_RETURN :=  11;
constant ONB_RESULT_OTHER:                     ONBOARD_RETURN :=  63;

--
-- Type for setting the device address linked to an application id
--

type DEVICE_ADDRESS = STRING (20);

--
-- Type for setting the SAS name defined for SWOP and Response packets
--

type  SAS_NAME = STRING (20);

constant NULL_SAS_NAME : SAS_NAME := "";



constant NO_DATA_LENGTH_LIMIT : INTEGER := 4081;
-- Remark:
--   maximum length of data field + 1;

type DATA_LENGTH_RANGE = INTEGER (0 .. NO_DATA_LENGTH_LIMIT);


--
-- type for setting the packet type of a swop command
--

constant CCSDS_DEFAULT_PACKET           : INTEGER :=  0;
constant CCSDS_MEMORY_DUMP_PACKET       : INTEGER :=  1;
constant CCSDS_DATA_SEGMENT_PACKET      : INTEGER :=  2;
constant CCSDS_ESSENTIAL_HK_PACKET      : INTEGER :=  3;
constant CCSDS_SYSTEM_HK_PACKET         : INTEGER :=  4;
constant CCSDS_PAYLOAD_HK_PACKET        : INTEGER :=  5;
constant CCSDS_SCIENCE_PACKET           : INTEGER :=  6;
constant CCSDS_SSMB_ANCILLIARY_PACKET   : INTEGER :=  7;
constant CCSDS_ESSENTIAL_COMMAND_PACKET : INTEGER :=  8;
constant CCSDS_SYSTEM_COMMAND_PACKET    : INTEGER :=  9;
constant CCSDS_PAYLOAD_COMMAND_PACKET   : INTEGER := 10;
constant CCSDS_MEMORY_LOAD_PACKET       : INTEGER := 11;
constant CCSDS_RESPONSE_PACKET          : INTEGER := 12;
constant CCSDS_REPORT_PACKET            : INTEGER := 13;
constant CCSDS_EXCEPTION_PACKET         : INTEGER := 14;
constant CCSDS_ACKNOWLEDGE_PACKET       : INTEGER := 15;

type CCSDS_PACKET_TYPE
  = INTEGER (CCSDS_DEFAULT_PACKET .. CCSDS_ACKNOWLEDGE_PACKET);


------------------------------------------------------------------------
-- Onboard Software Commanding / Flight Application Control
------------------------------------------------------------------------

1:  guarded procedure SET_CCSDS_END_POINT
      (in  GROUND_NODE : END_POINT;
       out STATUS      : UCL_RETURN);

-- Description:
--   Sets up the CCSDS ground destination (End Point), which is simulated by
--   the test node. From this end  point, the addressing parameter for the
--   CCSDS packets are derived when sending commands to onboard (e.g.
--   sw-commands or FLAP execution requests) and receiving responses.
-- Parameters:
--   IN:
--     GROUND_NODE: pathname of the CCSDS End Point which shall be simulated
--                  by the test node.
--   OUT:
--     STATUS:      UCL return status
--       OK                          The procedure was successful
--       INVALID_NODE_NAME           GROUND_NODE refers to an unknown item
--       INVALID_TESTNODE_MODE       The test node is not in the right mode
--                                   (NORMAL,SIMULATION)
--       RUNTIME_ERROR               Unexpected error

2:  guarded procedure ISSUE_SW_COMMAND
      (guarded in  SW_CMD          : SW_COMMAND ();
               in  ONBOARD_NODE    : END_POINT;
               in  GROUND_NODE     : END_POINT := \\;
               in  PRIO            : PRIORITY := LOW;
               in  TIMEOUT         : DURATION := default_sw_command_timeout;
               in  MAX_DATA_LENGTH : DATA_LENGTH_RANGE := NO_DATA_LENGTH_LIMIT;
               out RECEIVED_DATA_LENGTH : DATA_LENGTH_RANGE;
               out TRANSACTION_ID  : INTEGER;
               out RESULT          : ONBOARD_RETURN;
               out STATUS          : UCL_RETURN);
```

Copyright per DIN 34

```
-- Description:
--   Build and send a SWOP Telecommand.
--   Waits for the response packet and returns the SWOP Telecommand out
--   parameter (if any).
-- Parameters:
--   IN:
--     SW_CMD : name of the command to be executed onboard, followed by its
--              list of actual parameters (if any)
--     ONBOARD_NODE : Onboard node (CCSDS_End_Point) where the command is
--              sent to.
--     GROUND_NODE : Ground node which is simulated by CGS and where the
--              response has to be returned to.  If "\\" is given, the value
--              defined by SET_CCSDS_END_POINT is applicable.
--     PRIO: indicates the priority at which the SW-Command is to be executed
--              within the Ground System.
--     TIMEOUT: the maximum time to wait for acknowledgement or response packet
--              of this command.
--
--       Note: When setting the TIMEOUT to zero, the SW command as a CCSDS
--             packet will be sent to SAS, without requesting a response
--             packet. In this case, the timout for the acknowledge of the GDU
--             is set to a default, defined in the CGS configuration file as
--             parameter TES.KERNEL.SW_CMDER.ISSUE_TIMEOUT_WHEN_NO_DELAY.
--            (see $CGS_HOME/etc/configuration.xml)
--
--     MAX_DATA_LENGTH: the maximum length in bytes of the packet data field
--              (without checksum and headers). The data are truncated whenever
--              the parameters imply a longer data field. as follows: For
--              INTEGER the MSB of the integer and for STRING the maximum
--              length will be written, string length is set accordingly to the
--              truncated part. For other types, no data will be written.
--   OUT:
--     RECEIVED_DATA_LENGTH : the length in bytes of the out parameter part
--              of the resonse packet (without checksum, headers, transaction
--              id, onboard result). In case the received response packet is
--              shorter than requested the remaining out parameters shall be
--              set to their type respective null value (ie.
--
--                  type          | null value
--                  --------------|--------------
--                  INTEGER       |  0
--                  REAL          |  0.0,
--                  TIME          |  ~:~,
--                  PATHNAME      |  \\,
--                  STRING        |  "",
--                  STATECODE     |  $OTHER
--
--              The STATUS will be set to RESPONSE_TOO_SHORT in this case. If
--              a parameter cannot be filled completely then the parameter is
--              also set to the null values as described above, except for
--              INTEGER the remaining bytes are used as MSB of the integer and
--              STRING which are truncated, ie. length is set to truncated
--              string length.
--     TRANSACTION_ID:  Returns the ID as received from the response packet:
--              the primary header of the CCSDS uplink packet without the
--              length field
--     RESULT:  Returns the status of the execution as received from onboard
--              (returned in response packet)
--     STATUS:  UCL return status
--        OK                       The procedure was successful
--        BUSY                     The system (test node) is currently busy with
--                                 another SW command, try again later if necessary
--        INVALID_ITEM_NAME        The name of the SW Command is unknown
--        INVALID_NODE_NAME        The ID in ONBOARD_NODE or GROUND_NODE is
--                                 unknown
--        INVALID_TESTNODE_MODE    The test node is not in the right mode
--        ITEM_IS_DISABLED         The SW command is disabled
--        TIMEOUT                  There was no response from the SAS or
--                                 from the onboard system (response packet)
--        NO_GDU_SERVICE           The SAS did not announce the GDU Service
--        NO_ADU_SERVICE           The SAS did not announce the ADU Service
--        INVALID_APPLICATION_NAME The SAS is not known
--        APPLICATION_NOT_READY    The SAS is not connected
--        APPLICATION_NACK         There was a negative response from the SAS
--        INVALID TIME             The timetag given in the GDU header or
--                                 CCSDS header was not correct
--        PARAMETER_ERROR          When processing the parameter list of the
--                                 SW command an error occurred
--        COMMUNICATION_ERROR      problem in the communication to the SAS
--        RUNTIME_ERROR            error during the interpretation of the
--                                 UCL Intermediate Code
--        RESPONSE_TOO_SHORT       Response packet data length was shorter than
--                                 expected.
--        MESSAGE_TOO_BIG          Parameter list too big.
--        OPERATION_ABORTED        Aborted (eg. through ABORT_AP, STOP_TES)
--        INVALID_ADU_TYPE         ADU of wrong type (e.g. not CCSDS packet)
--                                 has been received as response
--        PROTOCOL_ERROR           Incorrect length of response packet or incorrect
--                                 checksum in case when the checksum is checked, see
--                                 configuration parameters
--                                 TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM and
```

Copyright per DIN 34

```
--                              TES.KERNEL.DATA_PROCESSOR.ADU.PROCESS_ON_INCORRECT_CHECKSUM

3:  guarded procedure ENABLE_SW_COMMAND
      (in  SW_CMD : SW_COMMAND;
       out STATUS : UCL_RETURN);

-- Description:
--   Enables a SWOP Command for execution
-- Parameters:
--   IN:
--     SW_CMD: name of the SW-command to enabled
--   OUT:
--     STATUS:       UCL return status
--       OK                  The procedure was successful
--                           This status is also returned if the item was
--                           already enabled
--      INVALID_ITEM_NAME     The name of the SW Command is unknown
--      INVALID_TESTNODE_MODE The test node is not in the right mode
--      RUNTIME_ERROR         Unexpected error

4:  guarded procedure DISABLE_SW_COMMAND
      (in  SW_CMD : SW_COMMAND;
       out STATUS : UCL_RETURN);

-- Description:
--   Disables a SWOP Command for execution
--   (i.e. ISSUE_SW_COMMAND command will be rejected for it)
-- Parameters:
--   IN:
--     SW_CMD: name of the SW-command to disabled
--   OUT:
--     STATUS:       UCL return status
--       OK                  The procedure was successful
--                           This status is also returned if the item was
--                           already disabled
--      INVALID_ITEM_NAME     The name of the SW Command is unknown
--      INVALID_TESTNODE_MODE The test node is not in the right mode
--      RUNTIME_ERROR         Unexpected error

5:  guarded procedure EXECUTE_FLAP
      (guarded in  FLAP                   : FLAP_NAME ();
               in  ONBOARD_RECEPTION_NODE : END_POINT;
               in  ONBOARD_EXECUTION_NODE : INSTANCE := \\;
               in  GROUND_NODE            : END_POINT := \\;
               in  GROUND_PRIO            : PRIORITY := LOW;
               in  ONBOARD_PRIO : ONBOARD_PRIORITY := default_onboard_priority;
               in  TIME_TAG               : TIME    := ~:~;
               in  TIMEOUT                : DURATION := default_flap_timeout;
               out TRANSACTION_ID         : INTEGER;
               out RESULT                 : ONBOARD_RETURN;
               out FLAP_EXEC_STATUS       : FLAP_RETURN;
               out ID                     : FLAP_ID;
               out STATUS                 : UCL_RETURN);

-- Description:
--   Builds a Telecommand – CCSDS Protocol Data Unit (PDU) – and initiates
--   the uplink of the command. Onboard the command instantiates and starts
--   a flight application program (FLAP).
--   Waits for the response packet and returns the parameter (packet is
--   returned after initiation of the FLAP)
--
--   Note: The carrier SWOP command should be defined as CGS configuration
--         file as parameter TES.KERNEL.SW_CMDER.EXEC_FLAP_SWOP_PATHNAME.
--         (see $CGS_HOME/etc/configuration.xml)
--
-- Parameters:
--   IN:
--     FLAP: name of the FLAP to be started, followed by its list of actual
--           parameters (if any)
--     ONBOARD_RECEPTION_NODE: onboard node (CCSDS End Point) to which the
--           command is to be sent to
--     ONBOARD_EXECUTION_NODE: onboard node (Instance), on which the FLAP shall
--           be executed.
--     GROUND_NODE: Ground node which is simulated by CGS and where the
--           response has to be returned to.  If "\\" is given, the value
--           defined by SET_CCSDS_END_POINT is applicable.
--     GROUND_PRIO: indicates the priority at which the command is to be sent
--           in the Ground System
--     ONBOARD_PRIO: indicates the priority at which the FLAP is to be executed
--           in the onboard system
--     TIME_TAG: indicates the time when to execute the FLAP.
--           The default value (~:~) indicates immediate execution (timetag is
--           set to 0 in the uplink packet). The time value is processed by the
--           onboard system
--     TIMEOUT: the maximum time to wait for acknowledgement or response packet
--           of this command.
--   OUT:
--     STATUS:       UCL return status
--        OK                     The procedure was successful
--        BUSY                   The system (test node) is currently busy
--                               with another SW command, try again later if
```

Copyright per DIN 34

```
--                                  necessary
--        INVALID_ITEM_NAME         The name of the FLAP is unknown
--        INVALID_NODE_NAME         The ID in ONBOARD_NODE or GROUND_NODE is
--                                  unknown
--        INVALID_TESTNODE_MODE     The test node isn't in right mode
--        ITEM_IS_DISABLED          The SW command defined for EXEC_FLAP is
--                                  disabled
--        TIMEOUT                   There was no response from the SAS or
--                                  from the onboard system (response packet)
--        NO_GDU_SERVICE            The SAS did not announce the GDU Service
--        NO_ADU_SERVICE            The SAS did not announce the ADU Service
--        INVALID_APPLICATION_NAME  The SAS is not known
--        APPLICATION_NOT_READY     The SAS is not connected
--        APPLICATION_NACK          There was a negative response from the SAS
--        INVALID TIME              The timetag given in the GDU header or
--                                  CCSDS header was not correct
--        COMMUNICATION_ERROR       There was a problem in the communication to
--                                  the SAS
--        SW_COMMAND_UNDEFINED      The SW Command, whose name/SID is defined
--                                  in the Configuration File, is not found in
--                                  the SW Command table, or it is even
--                                  not defined in the Configuration File
--        PARAMETER_ERROR           When processing the parameter list of the
--                                  FLAP command an error occurred
--        RUNTIME_ERROR             There was an error during the
--                                  interpretation of the UCL Intermediate Code
--        MESSAGE_TOO_BIG           Parameters to be put into CCSDS packet are
--                                  too big
--        PROTOCOL_ERROR            Incorrect length of response packet or
--                                  incorrect checksum in case when the checksum
--                                  is checked, see configuration parameters
--                                  TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM and
--                                  TES.KERNEL.DATA_PROCESSOR.ADU.PROCESS_ON_INCORRECT_CHECKSUM

6:  guarded procedure EXECWAIT_FLAP
      (in  FLAP                   : FLAP_NAME ();
       in  ONBOARD_RECEPTION_NODE : END_POINT;
       in  ONBOARD_EXECUTION_NODE : END_POINT := \\;
       in  GROUND_NODE            : END_POINT := \\;
       in  GROUND_PRIO            : PRIORITY := LOW;
       in  ONBOARD_PRIO : ONBOARD_PRIORITY := default_onboard_priority;
       in  TIME_TAG               : TIME    := ~:~;
       in  TIMEOUT                : DURATION := default_flap_timeout;
       out TRANSACTION_ID         : INTEGER;
       out RESULT                 : ONBOARD_RETURN;
       out ID                     : FLAP_ID;
       out FLAP_EXEC_STATUS       : FLAP_RETURN;
       out STATUS                 : UCL_RETURN);

-- Description:
--   this procedure is obsolete

7:  guarded procedure SET_DEVICE_ADDRESS
      (in SOURCE      : END_POINT;
       in DESTINATION : END_POINT;
       in ADDRESS     : DEVICE_ADDRESS;
       out STATUS     : UCL_RETURN);

-- Description:
--   Set a new device address for the application id identified by the source
--   node and the destination node. This device address will then be set in
--   the physical address of the SWOP commands and response packets that are
--   generated with that application id.
-- Parameters:
--   IN:
--     SOURCE : name of the source end point for identifying the application
--              id.
--     DESTINATION : name of the destination end point for identifying the
--               application id.
--     ADDRESS : string (up to 20 characters) to be set in the device_address
--               field of the physical_address.
--   OUT:
--     STATUS:       UCL return status
--        OK                     The procedure was successful
--        INVALID_NODE_NAME      There is no application id defined for
--                               the end point pair (SOURCE, DESTINATION)
--        RUNTIME_ERROR          There was an error during the
--                               interpretation of the UCL Intermediate Code

8:  guarded procedure ROUTE_SWOP_TO_SAS
      (in ITEM    :  SW_COMMANDS_AND_RESPONSES;
       in SAS     : SAS_NAME;
       in OLD_SAS : SAS_NAME :=  NULL_SAS_NAME;
       out STATUS : UCL_RETURN);

-- Description:
--   Routes all items specified in ITEM  which have the OLD_SAS_NAME assigned,
--   to the new SAS specified by SAS_NAME.
-- Parameters:
--   IN:
--     ITEM : The item to be re-defined in the local memory of the test node.
```

Copyright per DIN 34

```
--            ITEM can be
--               a single enditem of type SWOP_COMMAND or RESPONSE_PACKET
--               an incomplete pathname pointing to a virtual node, thus
--               redefining all SWOP_COMMAND or RESPONSE_PACKET in that subtree
--     SAS:  The short name of the SAS (Specific Application Software) to be
--           assigned to the ITEM. The SAS must run on the local test node.
--     OLD_SAS:  The short name of the SAS currently specified for item.
--           Only those items are taken into account for the route
--           change, which match this SAS name. If OLD_SAS is equal to
--           NULL_SAS_NAME, all enditems specified in ITEM are changed,
--           regardless of their current SAS name.
--   OUT:
--     STATUS:       UCL return status

 9:  procedure GET_CCSDS_PACKET_TYPE
       (in  SW_CMD      : SW_COMMAND;
        out PACKET_TYPE : CCSDS_PACKET_TYPE;
        out STATUS      : UCL_RETURN);

 -- Description:
 --   Gets the CCSDS packet type of a SW command.
 -- Parameters:
 --   IN:
 --     SW_CMD : name of the SW-command
 --   OUT:
 --     PACKET_TYPE:  type of the CCSDS packet (see constant definitions)
 --     STATUS:       UCL return status
 --        OK                      The procedure was successful
 --        INVALID_ITEM_NAME       SWOP command is unknown
 --        RUNTIME_ERROR           There was an error during the
 --                                interpretation of the UCL Intermediate Code

10:  guarded procedure SET_CCSDS_PACKET_TYPE
       (in  SW_CMD      : SW_COMMAND;
        in  PACKET_TYPE : CCSDS_PACKET_TYPE;
        out STATUS      : UCL_RETURN);


 -- Description:
 --   Sets the CCSDS packet type of a SW command.
 -- Parameters:
 --   IN:
 --     SW_CMD :      name of the SW-command
 --     PACKET_TYPE:  type of the CCSDS packet (see constant definitions)
 --   OUT:
 --     STATUS:       UCL return status
 --        OK                      The procedure was successful
 --        INVALID_ITEM_NAME       SWOP command is unknown
 --        RUNTIME_ERROR           There was an error during the
 --                                interpretation of the UCL Intermediate Code

11:  procedure BUILD_SW_COMMAND
       (guarded in  SW_CMD          : SW_COMMAND ();
                 in  ONBOARD_NODE    : END_POINT;
                 in  GROUND_NODE     : END_POINT := \\;
                 in  MAX_DATA_LENGTH : DATA_LENGTH_RANGE := NO_DATA_LENGTH_LIMIT;
                 out PACKET          : ARRAY OF INTEGER;
                 out STATUS          : UCL_RETURN);

 -- Description:
 --   Builds a raw CCSDS packet from a SW Command definition defined in
 --   the MDB
 --
 --   Note: the sequence counter within the primary header of the returned
 --         packet is set to 0
 --         if the returned packet does not fit into the provided parameter
 --         PACKET (ARRAY OF INTEGER), it will be truncated and the status
 --         INSUFFICIENT_PACKET_SIZE (63) will be returned.
 --
 -- Parameters:
 --   IN:
 --
 --   OUT:
 --     STATUS:       UCL return status


12:  guarded procedure ABORT_SW_COMMAND
       (in  SW_CMD          : SW_COMMAND;
        out STATUS          : UCL_RETURN);

 -- Description:
 --   Aborts the command SW_CMD if still waiting on response packet
 -- Parameters:
 --   IN:
 --
 --   OUT:
 --     STATUS:       UCL return status


end GROUND_TO_OB_LIB;
```

Copyright per DIN 34

I-5 UCL Ground System library: GROUND_VALUES

The GROUND_VALUES **library** contains standard ground commands used for access to measurement values, software variables and derived values.

library Id (Body_Id): 8

I-5.1 UCL GROUND_VALUES System library Specification

```
------------------------------------------------------------------
------
--
-- ****************************************
--  GROUND_VALUES UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to access acquired values in CGS
--     Must be compiled for ground and with Body Id = 8
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  :  GROUND_VALUES System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/ground_values_.ucl#10 $"
--*----------------------------------------------------------------
---
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
--
-------------------------------------------------------------------------

library GROUND_VALUES;

-------------------------------------------------------------------------
-- IMPORTS
-------------------------------------------------------------------------

  import GROUND_COMMON;

-------------------------------------------------------------------------
-- TYPES
-------------------------------------------------------------------------


  type CALIBRATION_TYPE = (NONE, POINT_PAIR, POLYNOM, IDENTICAL);


  type INTEG_VARIANT = (SIGNED,UNSIGNED);
  type REAL_VARIANT = (SINGLE,DOUBLE);

  type EXTENDED_MONITOR_STATUS =
    (DISABLED,
     NOT_ACQUIRED,
     NOMINAL,
     INTERRUPTED,        -- for delta limits only
     NON_NOMINAL,        -- for non-numeric items & delta limits
     OUT_OF_HIGH_LIMIT,  -- for numeric absolute limits
     OUT_OF_LOW_LIMIT);  -- dto.

  type LIMIT_VALUE = record
     case IS_INT : BOOLEAN
        when TRUE  :
          case INT_TYPE: INTEG_VARIANT
             when SIGNED:    I_VALUE : INTEGER;
             when UNSIGNED:  U_VALUE : UNSIGNED_INTEGER;
          end case;
        when FALSE :
          case R_TYPE: REAL_VARIANT
             when SINGLE:  R_VALUE : REAL;
             when DOUBLE:  L_VALUE : LONG_REAL;
          end case;
     end case;
  end record;

  type OPTIONAL_NUMERIC_LIMIT = record
     case DEFINED : BOOLEAN
        when FALSE :
        when TRUE  : VALUE : LIMIT_VALUE;
     end case;
  end record;

  type OPTIONAL_STATECODE = record
     case DEFINED : BOOLEAN
        when FALSE :
        when TRUE  : VALUE : STATECODE;
```

Copyright per DIN 34

```
   end case;
end record;

type OPTIONAL_VALUE_STRING = record
   case DEFINED : BOOLEAN
      when FALSE :
      when TRUE  : VALUE : VALUE_STRING;
   end case;
end record;

type NUMERIC_LIMITS = record
   HIGH,
   LOW,
   DELTA : OPTIONAL_NUMERIC_LIMIT;
end record;

type MONITOR_ITEM_CLASS = (NUMERIC, DISCRETE, BYTE_STREAM);

type FULL_MONITOR_STATUS = record
   case LIMITS_DEFINED : BOOLEAN
      when FALSE :
      when TRUE  :
        CURRENT_LIMIT_SET : INTEGER;
        NOMINAL_STATUS    : EXTENDED_MONITOR_STATUS;
        case ITEM_CLASS : MONITOR_ITEM_CLASS
           when NUMERIC     : DANGER_STATUS,
             DELTA_STATUS,
             DELTA_DANGER_STATUS :
             EXTENDED_MONITOR_STATUS;
             DANGER,
             NOMINAL: NUMERIC_LIMITS;
           when DISCRETE    : EXPECTED_STATE  : OPTIONAL_STATECODE;
           when BYTE_STREAM : EXPECTED_STRING : OPTIONAL_VALUE_STRING;
        end case;
        ALARM_COUNT : UNSIGNED_INTEGER;
   end case;
end record;

type CONDITION =
  (EQUAL, NOT_EQUAL, LESS, GREATER, LESS_EQUAL, GREATER_EQUAL, IN_RANGE);

type DISCRETE_CONDITION = CONDITION (EQUAL .. NOT_EQUAL);

type CONDITION_TYPE =
  (INTEGER_TYPE, REAL_TYPE, STATECODE_TYPE, BYTE_STREAM_TYPE);

type CONDITION_VAL = record
   case COND_TYPE : CONDITION_TYPE
      when INTEGER_TYPE     :    INTEGER_VALUE : INTEGER;
      when REAL_TYPE        :    REAL_VALUE : REAL;
      when STATECODE_TYPE   :    STATECODE_VALUE : STATECODE;
      when BYTE_STREAM_TYPE :    BYTE_STREAM_VALUE : VALUE_STRING;
   end case;
end record;

type ACTION_KIND =
  (PROCESSING,
   LIMIT,
   START_AP,
   MONITORING_ENABLE,
   MONITORING_DISABLE);

type ACTION_DESCRIPTION = record
   case KIND : ACTION_KIND
      when PROCESSING:
        PROCESS_ITEM : ACQUISITION_COLLECTION;
      when LIMIT:
        LIMIT_ITEM   : MONITOR_COLLECTION;
        LIMIT_SET    : LIMIT_SET_NUMBER;
      when START_AP:
        AP           : AP_NAME;
      when MONITORING_ENABLE:
        MON_LIMIT_ITEM : MONITOR_COLLECTION;
        MON_LIMIT_SET  : LIMIT_SET_NUMBER;
      when MONITORING_DISABLE :
        DISABLE_MON_LIMIT_ITEM : MONITOR_COLLECTION;
   end case;
end record;

type CONDITION_STATE = (IS_TRUE, IS_FALSE, IS_UNKNOWN);

type LIMIT_KIND =
   (NOMINAL_EXPECTED,           -- for expected statecode/bytestream only
    NOMINAL_HIGH,
    NOMINAL_LOW,
    NOMINAL_DELTA,
    DANGER_HIGH,
    DANGER_LOW,
    DANGER_DELTA);

type ACTION_NAME =
```

Copyright per DIN 34

```
        PATHNAME (UCL_AUTOMATED_PROCEDURE,
                  GDU_DESCRIPTION_LIST,
                  EGSE_ANALOG_STIMULUS,
                  EGSE_DISCRETE_STIMULUS,
                  EGSE_BINARY_PACKET,
                  EGSE_PREDEFINED_TC,
                  PUS_TC);

  -- used for output purposes
  type UCL_TEXT  = STRING(20);
  type T_RAW_VALUE_STRINGS = ARRAY (RAW_VALUE_TYPE) OF UCL_TEXT;
  type T_ENG_VALUE_STRINGS = ARRAY (ENG_VALUE_TYPE) OF UCL_TEXT;
  type T_ACQUISITION_STATUS_STRINGS = ARRAY (ACQUISITION_STATUS) OF UCL_TEXT;
  type T_ITEM_CLASS_STRINGS = ARRAY (MONITOR_ITEM_CLASS) OF UCL_TEXT;

  ------------------------------------------------------------------------
  -- CONSTANTS
  ------------------------------------------------------------------------

  constant DEFAULT_LIMIT_SET: LIMIT_SET_NUMBER := 0;

  constant ALL_CONDITIONS : INTEGER := 0;

  -- used for output purposes
  -- type RAW_VALUE_TYPE defined in ground_common
  constant RAW_VALUE_STRINGS: T_RAW_VALUE_STRINGS :=
    (”R_SIGNED”,
     ”R_UNSIGNED”,
     ”R_FLOAT”,
     ”R_DOUBLE_FLOAT”,
     ”R_BYTE_STREAM”);

  -- type ENG_VALUE_TYPE defined in ground_common
  constant ENG_VALUE_STRINGS: T_ENG_VALUE_STRINGS :=
    (”E_SIGNED”,
     ”E_UNSIGNED”,
     ”E_FLOAT”,
     ”E_DOUBLE_FLOAT”,
     ”E_STATE_CODE”,
     ”E_STRING”);

  constant ACQUISITION_STATUS_STRINGS: T_ACQUISITION_STATUS_STRINGS :=
    (”VALID”,
     ”NOT_ACQ”,
     ”NOT_RECVD”,
     ”NOT_MAINTAINED”,
     ”DISAB”,
     ”INVALID”,
     ”STATIC”);

  constant ITEM_CLASS_STRINGS: T_ITEM_CLASS_STRINGS :=
    (”NUMERIC”,
     ”DISCRETE”,
     ”BYTE_STREAM”);


------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  ----------------------------------------
  --  Calibration / Decalibration Functions
  --  are moved to calibration_lib
  ----------------------------------------


  ----------------------------------------
  --       Get Status and other attributes
  ----------------------------------------

  1:  procedure GET_ACQUISITION_STATUS
        (in  ITEM:                 MONITOR_ITEM_NAME;
         out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
         out STATUS:               UCL_RETURN);

  -- Description
  --   Returns the current acquisition status for an enditem. This operation
  --   can be called locally on the test node maintaining the enditem or
  --   remotely on another test node.
  -- Parameters:
  --   IN:
  --     ITEM : pathname of item to be checked.
  --   OUT:
  --     ENDITEM_ACQ_STATUS: returns the acquisition status of the enditem
  --                         (see GROUND_COMMON.ACQUISITION_STATUS)
  --     STATUS:             returns the UCL Return Code values :
  --       OK:                Information returned is valid
  --       RUNTIME_ERROR:     An exception occured when accessing the information
  --       INVALID_ITEM_NAME: Enditem is not known on the executing node

  2:  function STATUS_OF_ACQUISITION
```

Copyright per DIN 34

```
      (in  ITEM: MONITOR_ITEM_NAME) : ACQUISITION_STATUS;

-- Description
--   see above (GET_ACQUISITION_STATUS)
--     ACQUISITION_STATUS INVALID is returned for unkwown enditems


3:  procedure GET_ACQUISITION_TIME
      (in  ITEM:               ENDITEM_WITH_ENG_VALUE;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

-- Description
--   Return time of latest aquisition / update
--
--   Note: The acquisition time of a measurement is depend on
--         CGS configuration parameter TES.KERNEL.DATA_PROCESSOR.TIME_STAMP_IN_LT
--         and TES.KERNEL.DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_FROM_ADU.
--         (see $CGS_HOME/etc/configuration.xml)
-- Parameters:
--   IN:
--     ITEM : pathname of item to be checked.
--   OUT:
--     ENDITEM_ACQ_STATUS: returns the acquisition status of the enditem
--                         (see GROUND_COMMON.ACQUISITION_STATUS)
--     TIME_TAG:           return the time of latest acquisition/update
--     STATUS:             returns the UCL Return Code values :
--       OK:                Information returned is valid
--       RUNTIME_ERROR:     An exception occured when accessing the information
--       INVALID_ITEM_NAME: Enditem is not known on the executing node

4:  function ACQUISITION_TIME
      (in  ITEM: ENDITEM_WITH_ENG_VALUE) : TIME;

-- Description
--   see above (GET_ACQUISITION_TIME)
--     ~:~ is returned for unkwown enditems


5:  procedure GET_SAMPLE_COUNT
      (in  ITEM:               ENDITEM_WITH_ENG_VALUE;
       out COUNT:              INTEGER;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

-- Description
--   Gets the sample count for an enditem, i.e. the number of times a value
--   has been delivered for the enditem since start of testnode
-- Parameters:
--   IN:
--     ITEM : pathname of item to be checked.
--   OUT:
--     COUNT:              number of samples
--     ENDITEM_ACQ_STATUS: returns the acquisition status of the enditem
--     STATUS:             returns the UCL Return Code values :
--       OK                 : Information returned is valid
--       INVALID_ITEM_NAME  : Enditem is not known on the executing node

6:  function SAMPLE_COUNT
      (in  ITEM: ENDITEM_WITH_ENG_VALUE) : INTEGER;

-- Description
--   see above (GET_SAMPLE_COUNT)


7:  function NUMBER_VALUE_CHANGE
      (in  ITEM: ENDITEM_WITH_ENG_VALUE) : INTEGER;

-- Description
--   Return number of changes since the last start of testnode
-- Parameters:
--   IN:
--     ITEM : pathname of item to be checked.



-----------------------------------------
--  Get Type for Measurement Enditems
-----------------------------------------

8:  procedure GET_RAW_TYPE
      (in  ITEM:             ENDITEM_WITH_RAW_VALUE;
       out R_TYPE:           RAW_VALUE_TYPE;
       out RAW_SIZE_IN_BITS: INTEGER;
       out STATUS:           UCL_RETURN);

-- Description
--   Gets the raw value type for an enditem
-- Parameters:
--   IN:
```

Copyright per DIN 34

```
--     ITEM :  pathname of enditem
--   OUT:
--     R_TYPE:            raw value type of ITEM
--     RAW_SIZE_IN_BITS:  raw value size in bits
--     STATUS:            returns the UCL Return Code values:
--       OK
--       INVALID_ITEM_NAME

 9:  procedure GET_ENGINEERING_TYPE
       (in  ITEM:   ENDITEM_WITH_ENG_VALUE;
        out E_TYPE: ENG_VALUE_TYPE;
        out STATUS: UCL_RETURN);

-- Description
--   Gets the engineering value type for an enditem
-- Parameters:
--   IN:
--     ITEM :  pathname of enditem
--   OUT:
--     E_TYPE:            engineering value type of ITEM
--     STATUS:            returns the UCL Return Code values:
--       OK
--       INVALID_ITEM_NAME


------------------------------------------------------------------------------
--  Get Value and Status
------------------------------------------------------------------------------

--   Note: The acquisition time of a measurement is depend on
--         CGS configuration parameter TES.KERNEL.DATA_PROCESSOR.TIME_STAMP_IN_LT
--         and TES.KERNEL.DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_FROM_ADU.
--         (see $CGS_HOME/etc/configuration.xml)


10:  procedure GET_INTEGER
       (in  ITEM:               INTEGER_MONITOR_ITEM_NAME;
        out VALUE:              INTEGER;
        out TIME_TAG:           TIME;
        out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
        out STATUS:             UCL_RETURN);

-- Description
--   Returns the current value for an integer enditem. This operation can be
--   called locally on the test node maintaining the enditem or remotely on
--   another test node.
--
--   Note: This is the prefered way to find out the value of an enditem!
--         The direct access to an enditem, which is not acquired, stops
--         the execution of an Automated Procedure (x := \a\b\c;)
--
-- Parameters:
--   IN:
--     ITEM :  pathname of an integer enditem
--             (measurement or software variable or derived value).
--   OUT:
--     VALUE : the value of the enditem. In case the value is not VALID
--             (see ENDITEM_ACQ_STATUS), the last valid value will be
--             returned (or a default null value in case the value has never
--             been set)
--     TIME_TAG : the time tag (local time) of the last value. In case the
--             value is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the
--             last valid value will be returned (or a default null time in
--             case the value has never been set)
--     ENDITEM_ACQ_STATUS :  returns the acquisition status of item.
--     STATUS: returns the UCL Return Code values:
--       OK
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME

11:  procedure GET_UNSIGNED
       (in  ITEM:               UNSIGNED_MONITOR_ITEM_NAME;
        out VALUE:              UNSIGNED_INTEGER;
        out TIME_TAG:           TIME;
        out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
        out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER)

12:  procedure GET_FLOAT
       (in  ITEM:               FLOAT_MONITOR_ITEM_NAME;
        out VALUE:              REAL;
        out TIME_TAG:           TIME;
        out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
        out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER)

13:  procedure GET_DOUBLE_FLOAT
```

Copyright per DIN 34

```
      (in  ITEM:               DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       out VALUE:              LONG_REAL;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER)

14:  procedure GET_STATECODE
      (in  ITEM:               DISCRETE_MONITOR_ITEM_NAME;
       out VALUE:              STATECODE;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER)

15:  procedure GET_BYTE_STREAM
      (in  ITEM:               BYTE_STREAM_MONITOR_ITEM_NAME;
       out VALUE:              STRING;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER)

16:  procedure GET_INTEGER_RAW_VALUE
      (in  ITEM:               ENDITEM_WITH_SIGNED_RAW_VALUE;
       out VALUE:              INTEGER;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description:
 --     Returns the current raw value for an enditem having a raw value of type
 --     integer with the time tag of the latest update and the acquisition
 --     status. This operation can be called locally on the test node
 --     maintaining the enditem or remotely on another test node
 -- Parameters:
 --   IN:
 --     ITEM :  pathname of enditem (measurement).
 --   OUT:
 --     VALUE : the raw value of the enditem. In case the value is not VALID
 --             (see ENDITEM_ACQ_STATUS), the last valid value will be returned
 --             (or a default null value in case the value has never been set)
 --     TIME_TAG : the time tag (local time) of the last value. In case the value
 --             is not VALID (see ENDITEM_ACQ_STATUS) the time tag of the last
 --             valid value will be returned (or a default null time in case the
 --             value has never been set)
 --     ENDITEM_ACQ_STATUS :  returns the acquisition status of item.
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME

17:  function INTEGER_RAW_VALUE
      (in  ITEM: ENDITEM_WITH_SIGNED_RAW_VALUE) : INTEGER;

 -- Description:
 --     Returns the current raw value for an enditem having a raw value of type
 --     integer. This operation can be called locally on the test node
 --     maintaining the enditem or remotely on another test node
 -- Parameters:
 --   IN:
 --     ITEM :  pathname of enditem (measurement).

18:  procedure GET_UNSIGNED_RAW_VALUE
      (in  ITEM:               ENDITEM_WITH_UNSIGNED_RAW_VALUE;
       out VALUE:              UNSIGNED_INTEGER;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER_RAW_VALUE)

19:  function UNSIGNED_RAW_VALUE
      (ITEM: ENDITEM_WITH_UNSIGNED_RAW_VALUE) : UNSIGNED_INTEGER;

 -- Description
 --   see above (INTEGER_RAW_VALUE)

20:  procedure GET_FLOAT_RAW_VALUE
      (in  ITEM:               ENDITEM_WITH_FLOAT_RAW_VALUE;
       out VALUE:              REAL;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);
```

Copyright per DIN 34

```
 -- Description
 --   see above (GET_INTEGER_RAW_VALUE)

21:  function FLOAT_RAW_VALUE
      (ITEM: ENDITEM_WITH_FLOAT_RAW_VALUE): REAL;

 -- Description
 --   see above (INTEGER_RAW_VALUE)

22:  procedure GET_DOUBLE_FLOAT_RAW_VALUE
      (in  ITEM:               ENDITEM_WITH_DOUBLE_FLOAT_RAW_VALUE;
       out VALUE:              LONG_REAL;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER_RAW_VALUE)

23:  function DOUBLE_FLOAT_RAW_VALUE
      (ITEM: ENDITEM_WITH_DOUBLE_FLOAT_RAW_VALUE): LONG_REAL;

 -- Description
 --   see above (INTEGER_RAW_VALUE)

24:  procedure GET_BYTE_STREAM_RAW_VALUE
      (in  ITEM:               ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       out VALUE:              STRING OF BYTE;
       out TIME_TAG:           TIME;
       out ENDITEM_ACQ_STATUS: ACQUISITION_STATUS;
       out STATUS:             UCL_RETURN);

 -- Description
 --   see above (GET_INTEGER_RAW_VALUE)

 ------------------------------------------------------------------------------
 --  Set Value for SW variable
 ------------------------------------------------------------------------------

 -- Description:
 --   Set a value for a defined SW variable.
 --
 --   Note: The SET_<TYPE> functions are valid for SW variables only.
 --
 -- Parameters:
 --   IN:
 --     ITEM  : pathname of an enditem
 --     VALUE : value, ITEM set to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK            - setting was OK
 --       ITEM_UNKNOWN  - item not known
 --       NOT_OK        - no SW variable,
 --                       - range error
 --                       - type error
 --       RUNTIME_ERROR - unexpected error

95: procedure SET_INTEGER
      (in ITEM:    PATHNAME;
       in VALUE:   INTEGER;
       out STATUS: UCL_RETURN);


 -- Description
 --   see above (general decription)

96: procedure SET_UNSIGNED
      (in ITEM:    PATHNAME;
       in VALUE:   UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);


 -- Description
 --   see above (general decription)

97: procedure SET_FLOAT
      (in ITEM:    PATHNAME;
       in VALUE:   REAL;
       out STATUS: UCL_RETURN);


 -- Description
 --   see above (general decription)

98: procedure SET_DOUBLE_FLOAT
      (in ITEM:    PATHNAME;
       in VALUE:   LONG_REAL;
       out STATUS: UCL_RETURN);
```

Copyright per DIN 34

```
  -- Description
  --   see above (general decription)

 99: procedure SET_STATECODE
       (in ITEM:    PATHNAME;
        in VALUE:   STATECODE;
        out STATUS: UCL_RETURN);


  -- Description
  --   see above (general decription)

100: procedure SET_BYTE_STREAM
       (in ITEM:    PATHNAME;
        in VALUE:   STRING;
        out STATUS: UCL_RETURN);


  -- Description
  --   see above (general decription)


  ------------------------------------------------------------------------
  -- Monitoring
  ------------------------------------------------------------------------

 25:  guarded procedure ENABLE_MONITORING
       (in  ITEM:      MONITOR_COLLECTION;
        in  LIMIT_SET: LIMIT_SET_NUMBER := DEFAULT_LIMIT_SET;
        in  ADU:       ADU_NAME := \\;
        out STATUS:    UCL_RETURN);

  -- Description:
  --   Enables monitoring of enditems.
  -- Parameters:
  --   IN:
  --     ITEM : individual enditem or group of enditems to monitor.
  --
  --         ITEM               | enable monitoring for
  --         -------------------+-------------------------------------------
  --         MEASUREMENT        | single enditem
  --         VARIABLE           | single enditem
  --         DERIVED VALUE      | single enditem
  --         EGSE_MONITOR_LIST  | all enditems from list
  --         PUS_STRUCTURE_ID   | all enditems from list
  --         VIRTUAL            | all enditems under virtual node
  --         CDU                | all enditems under CDU node
  --
  --         Note: The acquisition for all measurements, which are not acquired,
  --               will be started automaticly!
  --
  --     LIMIT_SET : the limit set to be used. In case the value is not given
  --                 or the value given is DEFAULT_LIMIT_SET (0), the selected
  --                 limit set will be the currently selected limit set. Upon
  --                 loading of the database, the selected limit set is the
  --                 limit set 1. This can be changed by calling the operation
  --                 SET_LIMIT_SET.
  --     ADU : the ADU Descriptor to be applied. If ITEM is already acquired,
  --                 this parameter will be ignored. If there is only one ADU
  --                 Description defined for ITEM, the parameter may be omitted.
  --                 If more than one is defined and the parameter is omitted,
  --                 the command enables monitoring by starting acquisition for
  --                 all.
  --   OUT:
  --     STATUS returns the UCL Return Code values :
  --       OK
  --       INVALID_TESTNODE_MODE
  --       INVALID_ITEM_NAME
  --       NO_ADU_SERVICE
  --       INVALID_APPLICATION_NAME
  --       APPLICATION_NACK
  --       TIMEOUT
  --       INVALID_LIMIT_SET
  --       APPLICATION_NOT_READY

 26:  guarded procedure DISABLE_MONITORING
       (in  ITEM:   MONITOR_COLLECTION;
        out STATUS: UCL_RETURN);

  -- Description:
  --   Disables monitoring of enditems.
  -- Parameters:
  --   IN:
  --     ITEM: see above (ENABLE_MONITORING)
  --
  --         Note: The acquisition for all measurements, which are acquired,
  --               will NOT stopped automaticly!
  --
  --   OUT:
  --     STATUS returns the UCL Return Code values :
  --       OK
```

Copyright per DIN 34

```
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME

27:  guarded procedure VALUE_CHANGE_MONITORING
      (in  ITEM:    MONITOR_COLLECTION;
       in  SWITCH : ON_OFF;
       out STATUS:  UCL_RETURN);

 -- Description:
 --   Switch value change monitoring of enditem[s] on or off.
 --   Value change monitoring means, a monitoring action will be started,
 --   if the current value is not equal to the previous value.
 --   This procedure is independ to other monitoring activities.
 --
 --   Note: This procedure overwrites on local testnode the defaults
 --         loaded from MDB.
 --
 -- Parameters:
 --   IN:
 --     ITEM:   see above (ENABLE_MONITORING)
 --     SWITCH: ON  to enable the value change and
 --             OFF to disable it.
 --
 --       Note: The acquisition for all measurements, which are acquired,
 --             will NOT started and stopped automaticly!
 --
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME



28:  guarded procedure SET_HIGH_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the nominal high limit of an enditem in the currently selected
 --   limit set. If the item is an integer measurement or integer SW variable,
 --   LIMIT is rounded to the nearest integer.
 --
 --   Note: The nominal limits are inside the danger limit range (if defined).
 --
 --      low danger                                          high danger
 --            I-------------<-----------------><----------------------I
 --                     low nominal     high nominal
 --
 --   Note: If no limits are defined in the MDB, this routine sets the limits
 --         initial. The limits can be reused by the TSCV warm start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set high limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

29:  guarded procedure SET_INTEGER_HIGH_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_HIGH_LIMIT)

30:  guarded procedure SET_UNSIGNED_HIGH_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_HIGH_LIMIT)

31:  guarded procedure SET_DOUBLE_FLOAT_HIGH_LIMIT
      (in  ITEM:   DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_HIGH_LIMIT)

32:  guarded procedure SET_LOW_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
```

Copyright per DIN 34

```
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the nominal low limit of an enditem in the currently selected
 --   limit set. If the item is an integer measurement or integer SW variable,
 --   LIMIT is rounded to the nearest integer.
 --
 --   Note: If no limits are defined in the MDB, this routine sets the limits
 --         initial. The limits can be reused by the TSCV warm start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set low limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

33:  guarded procedure SET_INTEGER_LOW_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_LOW_LIMIT)

34:  guarded procedure SET_UNSIGNED_LOW_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_LOW_LIMIT)

35:  guarded procedure SET_DOUBLE_FLOAT_LOW_LIMIT
      (in  ITEM:   DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_LOW_LIMIT)

36:  guarded procedure SET_DELTA_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the nominal delta limit of an enditem in the currently selected
 --   limit set. If the item is an integer measurement or integer SW variable,
 --   LIMIT is rounded to the nearest integer.
 --
 --   Note: If no limits are defined in the MDB, this routine sets the limits
 --         initial. The limits can be reused by the TSCV warm start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set delta limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

37:  guarded procedure SET_INTEGER_DELTA_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DELTA_LIMIT)

38:  guarded procedure SET_UNSIGNED_DELTA_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DELTA_LIMIT)

39:  guarded procedure SET_DOUBLE_FLOAT_DELTA_LIMIT
      (in  ITEM:   DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);
```

Copyright per DIN 34

```
-- Description
--   see above (SET_DELTA_LIMIT)

40:  guarded procedure SET_EXCEPTION_COUNT
      (in  ITEM:    MONITOR_ITEM_NAME;
       in  N_COUNT: INTEGER;
       out STATUS:  UCL_RETURN);

-- Description:
--   Changes the exception count value of a measurement or SW variable.
--   The exception count is the number of nominal limit violations for
--   the occurrence of a nominal monitoring exception.
--
--   Note: If no limits are defined in the MDB, this routine sets the limits
--         initial. The limits can be reused by the TSCV warm start function.
--
-- Parameters:
--   IN:
--     ITEM :    name of item to have its exception count changed
--     N_COUNT : integer value to set exception count to
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME
--       INVALID_EXC_COUNT

41:  guarded procedure SET_EXPECTED_STATE
      (in  ITEM:   DISCRETE_MONITOR_ITEM_NAME;
       in  STATE:  STATECODE;
       out STATUS: UCL_RETURN);

-- Description:
--   Changes the expected state of a discrete measurement or SW variable.
--
--   Note: If no expected state is defined in the MDB, this routine sets the
--         expected state initial. The limits can be reused by the TSCV warm
--         start function.
--
-- Parameters:
--   IN:
--     ITEM :  name of item to change its expected state
--     STATE : state code for desired expected state of item
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME

42:  guarded procedure SET_EXPECTED_VALUE
      (in  ITEM:   BYTE_STREAM_MONITOR_ITEM_NAME;
       in  VALUE:  STRING;
       out STATUS: UCL_RETURN);

-- Description:
--   Changes the expected value of a byte stream measurement or SW variable.
--
--   Note: If no expected value is defined in the MDB, this routine sets the
--         expected value initial. The limits can be reused by the TSCV warm
--         start function.
--
-- Parameters:
--   IN:
--     ITEM :  name of item to change its expected value
--     VALUE : string containing desired expected value of item
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME

43:  guarded procedure SET_LIMIT_SET
      (in  ITEM:      MONITOR_COLLECTION;
       in  LIMIT_SET: LIMIT_SET_NUMBER;
       out STATUS:    UCL_RETURN);

-- Description:
--   Changes the current limit_set of a measurement or SW variable.  Any old
--   or other limit set is 'switched off' and the newly selected one is used
--   hereafter. In case a virtual path is specified, all measurements, SW
--   variables or derived values under this path get the new limit set number.
--   For items, where the new limit set is not defined, no change is made. The
--   same applies to monitoring lists.
-- Parameters:
--   IN:
--     see above (ENABLE_MONITORING)
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_TESTNODE_MODE
--       INVALID_ITEM_NAME
```

Copyright per DIN 34

```
--       INVALID_LIMIT_SET
--       INVALID_LIMIT

44:  guarded procedure SET_DANGER_HIGH_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the danger high limit of an enditem.
 --   If the item is an
 --         integer          LIMIT is rounded to the nearest integer.
 --         unsigned integer LIMIT is rounded to the nearest unsigned integer.
 --         real             LIMIT is rounded to the nearest real.
 --
 --   Note: If no danger limits defined in the MDB, this routine sets the danger
 --         limits initial. Without nominal limits, this routine fails. Define
 --         nominal limits before. The limits can be reused by the TSCV warm
 --         start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set danger high limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

45:  guarded procedure SET_INTEGER_DANGER_HIGH_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_HIGH_LIMIT)

46:  guarded procedure SET_UNSIGNED_DANGER_HIGH_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_HIGH_LIMIT)

47:  guarded procedure SET_FLOAT_DANGER_HIGH_LIMIT
      (in  ITEM:   FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_HIGH_LIMIT)

48:  guarded procedure SET_DANGER_LOW_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the danger low limit of an enditem.
 --   If the item is an
 --         integer          LIMIT is rounded to the nearest integer.
 --         unsigned integer LIMIT is rounded to the nearest unsigned integer.
 --         real             LIMIT is rounded to the nearest real.
 --
 --   Note: If no danger limits defined in the MDB, this routine sets the danger
 --         limits initial. Without nominal limits, this routine fails. Define
 --         nominal limits before. The limits can be reused by the TSCV warm
 --         start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set danger low limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

49:  guarded procedure SET_INTEGER_DANGER_LOW_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_LOW_LIMIT)
```

Copyright per DIN 34

```
50:  guarded procedure SET_UNSIGNED_DANGER_LOW_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_LOW_LIMIT)

51:  guarded procedure SET_FLOAT_DANGER_LOW_LIMIT
      (in  ITEM:   FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_LOW_LIMIT)

52:  guarded procedure SET_DANGER_DELTA_LIMIT
      (in  ITEM:   ANALOG_MONITOR_ITEM_NAME;
       in  LIMIT:  LONG_REAL;
       out STATUS: UCL_RETURN);

 -- Description:
 --   Changes the danger delta limit of an enditem.
 --   If the item is an
 --         integer          LIMIT is rounded to the nearest integer.
 --         unsigned integer LIMIT is rounded to the nearest unsigned integer.
 --         real             LIMIT is rounded to the nearest real.
 --
 --   Note: If no danger limits defined in the MDB, this routine sets the danger
 --         limits initial. Without nominal limits, this routine fails. Define
 --         nominal limits before. The limits can be reused by the TSCV warm
 --         start function.
 --
 -- Parameters:
 --   IN:
 --     ITEM :  name of item to change the limit
 --     LIMIT : value to set danger delta limit to
 --   OUT:
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_TESTNODE_MODE
 --       INVALID_ITEM_NAME
 --       INVALID_LIMIT

53:  guarded procedure SET_INTEGER_DANGER_DELTA_LIMIT
      (in  ITEM:   INTEGER_MONITOR_ITEM_NAME;
       in  LIMIT:  INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_DELTA_LIMIT)

54:  guarded procedure SET_UNSIGNED_DANGER_DELTA_LIMIT
      (in  ITEM:   UNSIGNED_MONITOR_ITEM_NAME;
       in  LIMIT:  UNSIGNED_INTEGER;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_DELTA_LIMIT)

55:  guarded procedure SET_FLOAT_DANGER_DELTA_LIMIT
      (in  ITEM:   FLOAT_MONITOR_ITEM_NAME;
       in  LIMIT:  REAL;
       out STATUS: UCL_RETURN);

 -- Description
 --   see above (SET_DANGER_DELTA_LIMIT)

56:  procedure GET_ENDITEM_MONITOR_STATUS
      (in  ITEM:              MONITOR_ITEM_NAME;
       out MONITORING_STATUS: MONITOR_STATUS;
       out STATUS:            UCL_RETURN);

 -- Description:
 --   Returns the current monitoring status for an enditem.
 -- Parameters:
 --   IN:
 --     ITEM  : name of enditem to monitor
 --   OUT:
 --     MONITORING_STATUS : monitoring status of ITEM
 --                         (see GROUND_COMMON.MONITOR_STATUS)
 --     STATUS returns the UCL Return Code values :
 --       OK
 --       INVALID_ITEM_NAME
 --       INVALID_TESTNODE_MODE


57:  procedure GET_FULL_ENDITEM_MONITOR_STATUS
      (in  ITEM:              MONITOR_ITEM_NAME;
       out MONITORING_STATUS: FULL_MONITOR_STATUS;
       out STATUS:            UCL_RETURN);
```

Copyright per DIN 34

```
-- Description:
--   Returns the current monitoring status and monitoring values for an
--   enditem.
-- Parameters:
--   IN:
--     ITEM  : name of enditem to monitor
--   OUT:
--     MONITORING_STATUS : detailed monitoring status of ITEM
--                         (see GROUND_COMMON.FULL_MONITOR_STATUS)
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_ITEM_NAME
--       INVALID_TESTNODE_MODE

58:  procedure GET_MONITORING_CLASS
      (in ENDITEM: MONITOR_ITEM_NAME;
       out CLASS:  MONITOR_ITEM_CLASS;
       out STATUS: UCL_RETURN);

-- Description:
--   Returns the monitoring class (numeric/discrete/bytestream) for an enditem.
-- Parameters:
--   IN:
--     ITEM :  pathname of enditem to monitor
--   OUT:
--     MONITORING_CLASS :  returns the enumeration describing the monitoring Class
--                         (see system library type description MONITOR_ITEM_CLASS)
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_ITEM_NAME
--       INVALID_TESTNODE_MODE


59:  function GET_MONITOR_GROUP_STATUS
      (in ITEM: MONITOR_COLLECTION) : MONITOR_STATUS;


-- Description:
--   Returns the summary of a monitoring status for a group of items or
--   the status of a single item. For summary calculation see
--   system library type description GROUND_COMMON.MONITOR_STATUS.
--
--     Note: The usage for groups of items in derived values is for cyclic
--           derived values only.
--
-- Parameters:
--   IN:
--     ITEM : item or group to be monitored
--            see above (ENABLE_MONITORING)

-------------------------------------------------------------------------
-- Engineering Value Logging
-------------------------------------------------------------------------

60:  procedure ENABLE_EVL
      (in  ITEM:   MONITOR_COLLECTION;
       out STATUS: UCL_RETURN);

-- Description:
--   Enables engineering value logging (EVL).
--   That means, that all engineering value samples of the items in
--   MONITOR_COLLECTION are logged for later evaluation.
--
--   Note: For each MDB enditem a default for EVL (Eng Val Log Control) can be
--         defined. Without entry in the MDB EVL is disabled by default.
--
-- Parameters:
--   IN:
--     ITEM : name of enditem(s) to be enabled
--            see above (ENABLE_MONITORING)
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_ITEM_NAME
--       INVALID_TESTNODE_MODE

61:  procedure DISABLE_EVL
      (in ITEM:    MONITOR_COLLECTION;
       out STATUS: UCL_RETURN);

-- Description:
--
-- Parameters:
--   IN:
--     ITEM : name of enditem(s) to be disabled
--            see above (ENABLE_MONITORING)
--   OUT:
--     STATUS returns the UCL Return Code values :
--       OK
--       INVALID_ITEM_NAME
```

Copyright per DIN 34

```
--       INVALID_TESTNODE_MODE

-------------------------------------------------------------------------
-- Conditions
-------------------------------------------------------------------------


-- Concept Description:
--
-- The conditional monitoring will be defined by linking 1 measurement or
-- software variable to one or more end items.
--
-- Every link can be described by
--
--   "when value of CONDITION_ITEM <<MATCHES> a CONDITION_VALUE"
--   "then <<PERFORM_AN_ACTION> on an ITEM"
--
-- where <<MATCHES> can be
--    EQUAL, NOT_EQUAL, LESS, GREATER, LESS OR EQUAL, GREATER OR EQUAL
--
--    (Note: the IN_RANGE alternative is currently not supported. It is included
--     in the definition of the CONDITION type only for MDB compatibility reasons )
--
-- where <<PERFORM_AN_ACTION> can be
--
--    "enable the processing of a measurement / a sw_variable / the
--                               measurements and sw variables of a
--                               monitoring list / the measurements of an
--                               ADU / the measurements and sw variables
--                               of a subtree"
--    "set a new limit set for a measurement / a sw_variable"
--    "start an AP"
--
-- When processing CONDITION_ITEM, all conditions linked to that item will
-- be analysed and for those that are true, the defined action will be
-- performed.
-- It is necessary to start the acquisition of CONDITION_ITEM to allow the
-- condition to be analysed.
--
-- For performance reasons, the conditions will only be analysed when the
-- value of the CONDITION_ITEM is modified, and not everytime the
-- CONDITION_ITEM is processed.
-- However, the when defining a condition, it will be checked immediately
-- if the CONDITION_ITEM has a valid value.
--
-- Note that the order of processing of the data from one ADU is the order of
-- the definition of the items in the ADU.
-- So if an adu contains an ITEM and its CONDITION_ITEM, to ensure that the
-- CONDITION_ITEM is processed before the ITEM, it must be defined in the
-- list of items of the ADU description before the ITEM.
--
--   Note: Please remember CGS configuration parameter
--
--           TES.KERNEL.DATA_PROCESSOR.USE_LIMIT_SET_NUMBER_AS_CONDITION
--
--         with following description:
--
--    (A) Allow or inhibit the usage of the MDB – defined Limit Set Number in
--         enditem condition description as additional condition for the Action
--         Type DISABLE_MONITORING only.
--
--         For all other conditon actions this parameter has no effect!
--
--         If this value is set to true, the condition is evaluated as follow:
--
--             condition false => do nothing
--
--             condition true  =>
--
--                condition Action Type = DISABLE_MONITORING =>
--
--                  ACTION ITEM REFERENCE = measurement, variable, derived:
--
--                    if condition Limit Set Number = current limit set
--                    OR condition Limit Set Number = 0 => disable_monitoring
--
--                  ACTION ITEM REFERENCE = monitoring list, virtual pathname:
--
--                    if condition Limit Set Number = current limit set (for at least
--                                                    one single enditem)
--                    OR condition Limit Set Number = 0 => disable_monitoring
--
--                    else                              => do nothing (*)
--
--                condition Action Type <> DISABLE_MONITORING => perform action
--
--         (*) In this case, the condition is not marked as triggered, because the
--                          limit set check is part of the condition.
--
--             Remember: The condition action is performed once only as long as
--                       the condition result is unchanged!
```

Copyright per DIN 34

```
--
--        Range:                 true/false
--        Recommended value:     false
--

62:  guarded procedure ENABLE_CONDITIONS
      (in  ITEM: ACQUISITION_COLLECTION;
       out STATUS: UCL_RETURN);

-- Description:
--   Will enable a condition or all conditions of
--   measurement(s), sw variable(s) and derived value(s).
-- Parameters:
--   IN:
--     ITEM : measurement(s), derived values(s) or software variable(s)
--            carrying the condition.
--   OUT:
--     STATUS : the return status of the operation
--       OK :                   operation successfull
--       INVALID_TESTNODE_MODE : the test node is not in executing mode
--       INVALID_ITEM_NAME :     ITEM is not managed locally
--       RUNTIME_ERROR     :     an unexpected error occurred

63:  guarded procedure DISABLE_CONDITIONS
      (in  ITEM: ACQUISITION_COLLECTION;
       out STATUS: UCL_RETURN);

-- Description
--   see above (ENABLE_CONDITIONS)

-------------------------------------------------------------------------
-- Set processing on condition
-------------------------------------------------------------------------

64:  guarded procedure ENABLE_ON_INTEGER
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

-- Description:
--   Will enable the processing of ITEM (measurement, software variable,
--   monitoring list or measurements of a subtree) when the CONDITION_CHECK
--   of the value of CONDITION_ITEM with CONDITION_VALUE is true (the
--   condition is true).
--
--   If the condition is false, this will disable the processing of ITEM.
--
--   By default, all enditems are enabled for processing. Conditions can be
--   defined to disable and re-enable the processing of given end-items.
--
--   When the processing is enabled, ITEM will be authorised for calibration
--   and monitoring. After ITEM has been enabled, it will be calibrated (if
--   its acquisition has been started) and monitored (if its monitoring has
--   been enabled) on reception of new values.
--
--   Enabling the processing of ITEM does not perform an "initial"
--   calibration / monitoring of ITEM in the case where it has already a value.
--
--   When the processing is disabled, ITEM will not be calibrated and not be
--   monitored, even if its acquisition has been requested and if its
--   monitoring is enabled.
--
--   CONDITION_ITEM must be maintained locally otherwise an error is generated.
--   When the condition is triggered, only those measurements and SW variables
--   identified by ITEM that are maintained locally will be processed.
--
--   It is not possible to set a condition where the ITEM to enable or disable
--   is the CONDITION_ITEM.
--   If ITEM is a group of enditems (i.e. monitoring list, ADU or incomplete
--   pathname) containing CONDITION_ITEM, the condition will not apply for
--   CONDITION_ITEM.
--
--   It is not possible to enable or disable the processing of a SW variable
--   that is linked to an HK data.
--
-- Parameters:
--   IN:
--     ITEM : item(s) to enable. It can be a measurement or software
--            variable or all measurements / sw variables contained in a
--            monitoring list or all measurements contained in an ADU or
--            all measurements / sw variables contained in a subtree.
--     CONDITION_ITEM : the measurement or software variable carrying the
--                      condition.
--     CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--     CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--     SINGLE_SHOT     : if true, the condition will be withdrawn after having
--                       being true.
```

Copyright per DIN 34

```
--   OUT:
--     CONDITION_REF   : a unique identifier for the condition
--     STATUS : the return status of the operation
--        OK:                 operation successfull
--        INVALID_TESTNODE_MODE : the test node is not in executing mode
--        INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
--        INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
--                            or ITEM is a SW variable linked to an HK data.
--                            or CONDITION_CHECK is IN_RANGE
--                            (not supported: see above)
--        RUNTIME_ERROR     : an unexpected error occurred

65:  guarded procedure ENABLE_ON_UNSIGNED_INTEGER
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : UNSIGNED_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : UNSIGNED_INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (ENABLE_ON_INTEGER)

66:  guarded procedure ENABLE_ON_FLOAT
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (ENABLE_ON_INTEGER)

67:  guarded procedure ENABLE_ON_DOUBLE_FLOAT
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : LONG_REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (ENABLE_ON_INTEGER)

68:  guarded procedure ENABLE_ON_STATECODE
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STATECODE;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (ENABLE_ON_INTEGER)

69:  guarded procedure ENABLE_ON_BYTE_STREAM
      (in  ITEM            : ACQUISITION_COLLECTION;
       in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STRING;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (ENABLE_ON_INTEGER)

 ------------------------------------------------------------------------
 -- Processing
 ------------------------------------------------------------------------

70:  guarded procedure SET_PROCESSING
      (in  ITEM   : ACQUISITION_COLLECTION;
       in  SWITCH : ON_OFF;
       out STATUS : UCL_RETURN);

 -- Description:
 --   Will enable or disable the processing of ITEM. This is overwriting
 --   an eventual action from a condition.
 --
 --     Note: Disabling of processing does not set the monitoring status to
 --           'disabled'. Thus the re-enabling of processing also re-enables
 --           monitoring, if monitoring was enabled before.
 --
 -- Parameters:
 --   IN:
```

Copyright per DIN 34

```
--     ITEM : item(s) to enable or disable. It can be a measurement or
--            software variable or all measurements / sw variables contained
--            in a monitoring list or all measurements contained in an ADU or
--            all measurements / sw variables contained in a subtree.
--     SWITCH : ON to enable the processing and OFF to disable the processing.
--   OUT:
--     STATUS : the return status of the operation
--        OK                : operation successfull
--        INVALID_TESTNODE_MODE : the test node is not in executing mode
--        INVALID_ITEM_NAME : ITEM is not managed locally
--        INVALID_PARAMETER : ITEM is a SW variable linked to an HK data.
--                            or CONDITION_CHECK is IN_RANGE (not supported: see above)
--        RUNTIME_ERROR     : an unexpected error occurred


71:  procedure GET_PROCESSING_STATE
      (in  ITEM   : MONITOR_ITEM_NAME;
       out STATE  : ON_OFF;
       out STATUS : UCL_RETURN);

 -- Description:
 --   Indicates if a measurement or software variabled is enabled or disabled
 --   for processing (as a result of a condition or of a call to SET_PROCESSING)
 -- Parameters:
 --   IN:
 --     ITEM : item to check. It can be a single measurement or software
 --            variable.
 --     STATE : ON if the processing is enabled and OFF if the processing is
 --     disabled.
 --   OUT:
 --     STATUS : the return status of the operation
 --        OK                : operation successfull
 --        INVALID_TESTNODE_MODE : the test node is not in executing mode
 --        INVALID_ITEM_NAME : ITEM is not managed locally
 --        RUNTIME_ERROR     : an unexpected error occurred


 ------------------------------------------------------------------------
 -- Set Limit on condition
 ------------------------------------------------------------------------

72:  guarded procedure SET_LIMIT_SET_ON_INTEGER
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description:
 --   Will set the LIMIT_SET for ITEM (measurement or software variable) when
 --   the CONDITION_CHECK of the value of CONDITION_ITEM with CONDITION_VALUE
 --   is true (the condition is true).
 --
 --   This will not perform an ”initial” monitoring of ITEM in the case where
 --   it has already a value. The monitoring will be performed on reception of
 --   the next value (in case the monitoring has been enabled).
 --   This can lead to a temporary ”inconsistent” output of the procedure
 --   GET_FULL_ENDITEM_MONITOR_STATUS (result of the monitoring with the old
 --   limit set with values of the new limit set might be inconsistent).
 --
 --   ITEM and CONDITION_ITEM must be maintained locally on the same node
 --   otherwise an error is generated.
 --
 --   It is not possible to set a condition where the ITEM identical to the
 --   CONDITION_ITEM.
 --
 --   The LIMIT_SET must be defined for ITEM, otherwise an error is generated
 --
 -- Parameters:
 --   IN:
 --     ITEM : item for which the limit set is to be switched. It can be a
 --            measurement or software variable.
 --     LIMIT_SET : the limit set number to select.
 --     CONDITION_ITEM : the measurement or software variable carrying the
 --                      condition.
 --     CONDITION_CHECK : the check to apply on CONDITION_ITEM.
 --     CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
 --     SINGLE_SHOT     : if true, the condition will be withdrawn after having
 --                       being true.
 --   OUT:
 --     CONDITION_REF   : a unique identifier for the condition
 --     STATUS : the return status of the operation
 --        OK                : operation successfull
 --        INVALID_TESTNODE_MODE : the test node is not in executing mode
 --        INVALID_ITEM_NAME : ITEM or CONDITION_ITEM are not managed locally
 --        INVALID_PARAMETER : ITEM and CONDITION_ITEM are identical
 --                            or CONDITION_CHECK is IN_RANGE (not supported: see above)
```

Copyright per DIN 34

```
--        INVALID_LIMIT_SET : the limit set is not defined for ITEM
--        RUNTIME_ERROR     : an unexpected error occurred

73:  guarded procedure SET_LIMIT_SET_ON_UNSIGNED_INTEGER
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : UNSIGNED_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : UNSIGNED_INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (SET_LIMIT_SET_ON_INTEGER)

74:  guarded procedure SET_LIMIT_SET_ON_FLOAT
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (SET_LIMIT_SET_ON_INTEGER)

75:  guarded procedure SET_LIMIT_SET_ON_DOUBLE_FLOAT
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : LONG_REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (SET_LIMIT_SET_ON_INTEGER)

76:  guarded procedure SET_LIMIT_SET_ON_STATECODE
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STATECODE;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (SET_LIMIT_SET_ON_INTEGER)

77:  guarded procedure SET_LIMIT_SET_ON_BYTE_STREAM
      (in  ITEM            : MONITOR_COLLECTION;
       in  LIMIT_SET       : LIMIT_SET_NUMBER;
       in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STRING;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (SET_LIMIT_SET_ON_INTEGER)

 ------------------------------------------------------------------------
 -- Start Automated Procedure on condition
 ------------------------------------------------------------------------

78:  guarded procedure START_AP_ON_INTEGER
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : INTEGER_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description:
 --   Will start an automated procedure when the CONDITION_CHECK of the value
 --   of CONDITION_ITEM with CONDITION_VALUE is true (the condition is true).
 --
 --   CONDITION_ITEM must be maintained locally on the node otherwise an
 --   error is generated.
 --
 --   The AP to start must be an AP without parameters, otherwise an error
 --   message is generated when the condition is triggered.
 --   When the condition is triggered, if the AP is already running, it will
```

Copyright per DIN 34

```
--   not be started and an error message will be generated.
--
-- Parameters:
--   IN:
--     AP : the name of the automated procedure to start. This must be an AP
--          without parameter or where all parameters have default values.
--     CONDITION_ITEM : the measurement or software variable carrying the
--                      condition.
--     CONDITION_CHECK : the check to apply on CONDITION_ITEM.
--     CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
--     SINGLE_SHOT     : if true, the condition will be withdrawn after having
--                       being true.
--   OUT:
--     CONDITION_REF   : a unique identifier for the condition
--     STATUS : the return status of the operation
--        OK                : operation successfull
--        INVALID_TESTNODE_MODE : the test node is not in executing mode
--        INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally
--        INVALID_PARAMETER : CONDITION_CHECK is IN_RANGE (not supported: see above)
--        RUNTIME_ERROR     : an unexpected error occurred


79:  guarded procedure START_AP_ON_UNSIGNED_INTEGER
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : UNSIGNED_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : UNSIGNED_INTEGER;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (START_AP_ON_INTEGER)

80:  guarded procedure START_AP_ON_FLOAT
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (START_AP_ON_INTEGER)

81:  guarded procedure START_AP_ON_DOUBLE_FLOAT
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : DOUBLE_FLOAT_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : CONDITION;
       in  CONDITION_VALUE : LONG_REAL;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (START_AP_ON_INTEGER)

82:  guarded procedure START_AP_ON_STATECODE
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : DISCRETE_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STATECODE;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (START_AP_ON_INTEGER)

83:  guarded procedure START_AP_ON_BYTE_STREAM
      (in  AP              : AP_NAME;
       in  CONDITION_ITEM  : BYTE_STREAM_MONITOR_ITEM_NAME;
       in  CONDITION_CHECK : DISCRETE_CONDITION;
       in  CONDITION_VALUE : STRING;
       in  SINGLE_SHOT     : BOOLEAN;
       out CONDITION_REF   : INTEGER;
       out STATUS          : UCL_RETURN);

 -- Description
 --   see above (START_AP_ON_INTEGER)


84:  guarded procedure WITHDRAW_CONDITION
      (in  CONDITION_ITEM : MONITOR_ITEM_NAME;
       in  CONDITION_REF  : INTEGER := ALL_CONDITIONS;
       out STATUS         : UCL_RETURN);


 -- Description
 --   Will remove a condition or all conditions of a CONDITION_ITEM.
```

Copyright per DIN 34

```
-- Parameters:
--   IN:
--     CONDITION_ITEM : the measurement or software variable carrying the
--                      condition.
--     CONDITION_REF   : the unique identifier of the condition to remove
--                       or the constant ALL_CONDITIONS to remove all conditions
--                       of CONDITION_ITEM.
--   OUT:
--     STATUS : the return status of the operation
--        OK               : operation successfull
--        INVALID_TESTNODE_MODE : the test node is not in executing mode
--        INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally or
--                            CONDITION_REF does not identify an existing
--                            condition
--        RUNTIME_ERROR     : an unexpected error occurred


85:  guarded procedure WITHDRAW_ALL_CONDITIONS
      (out STATUS : UCL_RETURN);

 -- Description
 --   Will remove all conditions defined on the test node.
 -- Parameters:
 --   OUT:
 --     STATUS : the return status of the operation
 --        OK               : operation successfull
 --        INVALID_TESTNODE_MODE : the test node is not in executing mode
 --        RUNTIME_ERROR     : an unexpected error occurred


 ------------------------------------------------------------------------
 -- MANAGE CONDITIONS
 ------------------------------------------------------------------------

86:  function NUMBER_OF_CONDITION_ITEMS : INTEGER;

 -- Description:
 --   Returns the number of items carrying conditions (CONDITION_ITEMS).
 --   This number is as well maintained as Housekeeping variable (and can
 --   be mapped to a software variable).



87:  procedure GET_CONDITION_ITEM
      (in  ITEM_NUMBER    : INTEGER;
       out CONDITION_ITEM : MONITOR_ITEM_NAME;
       out STATUS         : UCL_RETURN);

 -- Description:
 --   Provide the name of the CONDITION_ITEM identified by an index.
 --   To get all the CONDITION_ITEM's, this procedure has to be called
 --   with all numbers betwee 1 and the value of NUMBER_OF_CONDITION_ITEMS.
 --
 --   Note:
 --     it can occur that between the time where NUMBER_OF_CONDITION_ITEMS has
 --     been called and the time where GET_CONDITION_ITEM is called, the number
 --     of items carrying condition is modified (if new conditions are created
 --     or removed in parallel).
 --
 -- Parameters:
 --   IN:
 --     ITEM_NUMBER : index of the CONDITION_ITEM (between 1 and the value of
 --                   NUMBER_OF_CONDITION_ITEMS).
 --   OUT:
 --     CONDITION_ITEM : the measurement or software variable carrying the
 --                      condition.
 --     STATUS : the return status of the operation
 --        OK               : operation successfull
 --        INVALID_TESTNODE_MODE : the test node is not in executing mode
 --        INVALID_PARAMETER : There is no (more) CONDITION_ITEM at the index
 --                            ITEM_NUMBER (see above note).
 --        RUNTIME_ERROR     : an unexpected error occurred


88:  function NUMBER_CONDITIONS
      (in  CONDITION_ITEM : MONITOR_ITEM_NAME) :INTEGER;

 -- Description:
 --   Provide the number of conditions carried by a CONDITION_ITEM.
 -- Parameters:
 --   IN:
 --     CONDITION_ITEM : the name of the item for which the number of conditions
 --                      is to be returned. It must be managed locally, otherwise
 --                      the value 0 is returned.


89:  procedure GET_CONDITION
      (in  CONDITION_ITEM   : MONITOR_ITEM_NAME;
       in  CONDITION_NUMBER : INTEGER;
       out CONDITION_REF    : INTEGER;
```

Copyright per DIN 34

```
      out CONDITION_CHECK  : CONDITION;
      out CONDITION_VALUE  : CONDITION_VAL;
      out SINGLE_SHOT      : BOOLEAN;
      out STATE            : CONDITION_STATE;
      out ACTION           : ACTION_DESCRIPTION;
      out STATUS           : UCL_RETURN);

 -- Description:
 --   Provide the description of a condition.
 --
 --   Note:
 --     it can occur that between the time where GET_NUMBER_CONDITIONS has
 --     been called and the time where GET_CONDITION is called, the number
 --     of condition carried by CONDITION_ITEM is modified
 --     (if new conditions are created or removed in parallel).
 --
 -- Parameters:
 --   IN:
 --     CONDITION_ITEM : the name of the item carrying the condition.
 --     CONDITION_NUMBER : the index of the condition (between 1 and the value
 --                        of GET_NUMBER_CONDITIONS).
 --   OUT:
 --     CONDITION_REF : the unique identifier of the condition.
 --     CONDITION_CHECK : the check to apply on CONDITION_ITEM.
 --     CONDITION_VALUE : the value to use in the check with CONDITION_ITEM.
 --     SINGLE_SHOT : indicates if the condition is to be removed after having
 --                   being true.
 --     STATE : indicate the state of the condition. It can be
 --             IS_TRUE : the condition is true, the associated action has been
 --                       triggered
 --             IS_FALSE : the condition is false, the associated action has been
 --                        triggered (only for the enable/disable conditions)
 --             IS_UNKNOWN : the condition has been set but has not yet been
 --                          analysed (CONDITION_ITEM has not a valid value)
 --     ACTION : provide the description of the condition, i.e:
 --                - the sid of the item(s) for which the processing is to be
 --                  enabled
 --                - the sid of the item for which a new limit set is to be set
 --                  and the limit set number
 --                - the sid of an AP
 --     STATUS : the return status of the operation
 --        OK                : operation successfull
 --        INVALID_TESTNODE_MODE : the test node is not in executing mode
 --        INVALID_PARAMETER : There is no (more) condition at the index
 --                            CONDITION_NUMBER (see above note).
 --        INVALID_ITEM_NAME : CONDITION_ITEM is not managed locally.
 --        RUNTIME_ERROR     : an unexpected error occurred

 ------------------------------------------------------------------------
 -- Pathname / PUS Parameter Number Translation
 ------------------------------------------------------------------------

90:  function PATH_FOR_PUS_PARAMETER_NUMBER
      (in PUS_PARAMETER_NUMBER: UNSIGNED_INTEGER): PATHNAME;

91:  function PUS_PARAMETER_NUMBER
      (in ENDITEM_PATH: ENDITEM_WITH_RAW_VALUE): UNSIGNED_INTEGER;

 -- Description
 --   Translates a Pathname to a PUS Parameter Number (as used for onboard reference)
 --   and vice versa.
 --   Parameter Number value is derived from the attribute associated with measurements
 --   in the MDB
 --
 --   Note: functions will work only for enditems loaded to the local test node
 --         In case the PUS_PARAMETER_NUMBER is not known/found,
 --         PATH_FOR_PUS_PARAMETER_NUMBER will return the empty pathname "\\"
 --         In case the ENDITEM_PATH is not known/found, a 0 is returned.
 --

 ------------------------------------------------------------------------
 -- measurement source information
 ------------------------------------------------------------------------


92:  function MEASUREMENT_SOURCE_INFORMATION
      (in ENDITEM_PATH: ENDITEM_WITH_RAW_VALUE): INTEGER;


 -- Description
 --   Returns the source information for the specified measurement.
 --
 --   Note: Function will work only for enditems loaded to the local test node.
 --         The measurement will inherit the source information from ADU.
 --         Which kind of source information is used depends on CGS
 --         configuration parameter
 --         TES.KERNEL.DATA_PROCESSOR.ADU.USE_APID_FOR_TM_PACKET_IDENTIFICATION
 --         and TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID.
 --
 --   returns negative value (-1) if
 --           a)   the ENDITEM_PATH is not known / not found
```

Copyright per DIN 34

```
  --           b)   the measurement is depend on a non CCSDS-packet and
  --                should use the APID
  --           c)   no Source_Identifier for ADU is set
  --    returns (extended) APID of CCSDS-packet or Source_Identifier from ADU
  --

 93:  procedure SET_LIMIT_ACTION
       ( in  ITEM      : MONITOR_ITEM_NAME;
         in  ACTION    : ACTION_NAME;
         in  KIND      : LIMIT_KIND := NOMINAL_EXPECTED;
         in  EMERGENCY : BOOLEAN := TRUE;
         out STATUS    : UCL_RETURN);

  -- Description:
  --   Changes the limit action of an enditem.
  --
  --   Note: If no action is defined in the MDB, this routine sets the action
  --         initial. Without limits, this routine fails.
  --
  -- Parameters:
  --   IN:
  --     ITEM :   name of item to change the action
  --     ACTION : action, which shall performed in case of limit violation
  --              If ACTION = \\, the action will be deleted for this ITEM
  --     KIND :   kind of limit
  --     EMERGENCY: this parameter defines the priority of the monitoring AP
  --              and is only used, if ACTION is of type UCL_AUTOMATED_PROCEDURE.
  --              EMERGENCY = TRUE  => AP runs with emergency priority
  --              EMERGENCY = FALSE => AP runs with low priority
  --   OUT:
  --     STATUS returns the UCL Return Code values :
  --       OK
  --       INVALID_TESTNODE_MODE
  --       INVALID_ITEM_NAME
  --       INVALID_PARAMETER


 94:  procedure SET_LIMIT_MESSAGE
       ( in  ITEM    : MONITOR_ITEM_NAME;
         in  MESSAGE : USER_MESSAGE_NAME;
         in  KIND    : LIMIT_KIND := NOMINAL_EXPECTED;
         out STATUS  : UCL_RETURN);

  -- Description:
  --   Changes the limit message of an enditem.
  --
  --   Note: If no action is defined in the MDB, this routine sets the action
  --         initial. Without limits, this routine fails.
  --
  -- Parameters:
  --   IN:
  --     ITEM :   name of item to change the action
  --     MESSAGE: user message, which shall displayed in case of limit violation
  --              If MESSAGE = \\, the message will be deleted for this ITEM
  --     KIND :   kind of limit
  --   OUT:
  --     STATUS returns the UCL Return Code values :
  --       OK
  --       INVALID_TESTNODE_MODE
  --       INVALID_ITEM_NAME
  --       INVALID_PARAMETER

END GROUND_VALUES;
```

Copyright per DIN 34

I-6 UCL Ground System library: MATH_LIB

The MATH_LIB **library** contains standard mathematical operations for real **type** parameter.

library Id (Body_Id): 6

I-6.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  MATH_LIB UCL System Library
-- ****************************************
--
-- ABSTRACT
--     Defines Functions for basic mathematic operations
--     for Real type.
--     Must be compiled for ground and with Body Id = 6
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : MATH_LIB System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/math_lib_.ucl#5 $"
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------


library MATH_LIB;

------------------------------------------------------------------------


   --    GENERAL:
   --    --------
   --    1. The functions have their usual mathematical meanings. The Log function
   --       computes the logarithm to the given base; LogLn computes the natural
   --       logarithm. When the Cycle parameter is specified, the parameter X
   --       of the forward trigonometric functions (Sin2, Cos2, Tan2, and Cot2)
   --       and the results of the inverse trigonometric functions (Arcsin2,
   --       Arccos2, Arctan2, and Arccot2) are measured in units such that a
   --       full cycle of revolution has the given value; otherwise (Sin, Cos,
   --       Tan, Cot, Arcsin, Arccos, Arctan, Arccot,Sinh, Cosh, Tanh, Coth ,
   --       Arsinh, Arcosh, Artanh and Arcoth), they are measured in radians.
   --    2. The computed results of the mathematically multivalued functions are
   --       rendered single-valued by the following conventions, which are meant to
   --       imply the principal branch:
   --           1.The results of the Sqrt and Arcosh functions and that of the
   --             exponentiation operator are nonnegative.
   --           2.The result of the Arcsin function is in the quadrant containing the
   --             point (1.0, x), where x is the value of the parameter X. This
   --             quadrant is I or IV; thus, the range of the Arcsin function is
   --             approximately -Pi/2.0 to Pi/2.0 (-Cycle/4.0 to Cycle/4.0, if the
   --             parameter Cycle is specified).
   --           3.The result of the Arccos function is in the quadrant containing the
   --             point (x, 1.0), where x is the value of the parameter X. This
   --             quadrant is I or II; thus, the Arccos function ranges from 0.0 to
   --             approximately Pi (Cycle/2.0, if the parameter Cycle is specified).
   --           4.The results of the Arctan and Arccot functions are in the quadrant
   --             containing the point (x, y), where x and y are the values of the
   --             parameters X and Y, respectively. This may be any quadrant
   --             (I through IV) when the parameter X (resp., Y) of Arctan (resp.,
   --             Arccot) is specified, but it is restricted to quadrants I and IV
   --             (resp., I and II) when that parameter is omitted. Thus, the range
   --             when that parameter is specified is approximately -Pi to Pi
   --             (-Cycle/2.0 to Cycle/2.0, if the parameter Cycle is specified); when
   --             omitted, the range of Arctan (resp., Arccot) is that of
   --             Arcsin (resp., Arccos), as given above. When the point (x, y) lies
   --             on the negative x-axis, the result approximates Pi (resp., -Pi) when
   --             the sign of the parameter Y is positive (resp., negative)
   --
   --
   --       (In the case of the inverse trigonometric functions, in which a result
   --       lying on or near one of the axes may not be exactly representable, the
   --       approximation inherent in computing the result may place it
   --       in an adjacent quadrant, close to but on the wrong side of the axis.)
   --
   --    3. The random functions returns a value in integer range (Random_I)
   --       or in range 0.0 .. 1.0 (Random).
   --
   --    4. The function Last_Exception returns the exception of the last math lib
   --       function call. In case of Not_Exception the value returned by the last math
   --       lib function call was valid. In case of any other value the value returned
   --       by the last math lib function call was invalid.
   --
```

Copyright per DIN 34

```
   --     Dynamic Semantics:
   --     ------------------
   --     2.The exception Argument_Error (function Last_Exception) is raised,
   --       signaling a parameter value outside the domain of the corresponding
   --       mathematical function, in the following cases:
   --           1.by any forward or inverse trigonometric function with specified
   --             cycle, when the value of the parameter Cycle is zero or negative;
   --           2.by the Log function with specified base, when the value of the
   --             parameter Base is zero, one, or negative;
   --           3.by the Sqrt and Log functions, when the value of the parameter
   --             X is negative;
   --           4.by the exponentiation operator, when the value of the left
   --             operand is negative or when both operands have the value zero;
   --           5.by the Arcsin, Arccos, and Artanh functions, when the absolute
   --             value of the parameter X exceeds one;
   --           6.by the Arctan and Arccot functions, when the parameters X and Y
   --             both have the value zero;
   --           7.by the Arcosh function, when the value of the parameter X is
   --             less than one; and
   --           8.by the Arcoth function, when the absolute value of the parameter
   --             X is less than one.
   --
   --     1.The exception Constraint_Error (function Last_Exception) is raised,
   --       signaling a pole of the mathematical function (analogous to dividing
   --       by zero), in the following cases:
   --           1.by the Log, Cot, and Coth functions, when the value of the parameter
   --             X is zero;
   --           2.by the exponentiation operator, when the value of the left operand
   --             is zero and the value of the exponent is negative;
   --           3.by the Tan function with specified cycle, when the value of the
   --             parameter X is an odd multiple of the quarter cycle;
   --           4.by the Cot function with specified cycle, when the value of the
   --             parameter X is zero or a multiple of the half cycle; and
   --           5.by the Artanh and Arcoth functions, when the absolute value
   --             of the parameter X is one.
   --
   --       Constraint_Error can also be raised when a finite result overflows;
   --       this may occur for parameter values sufficiently near poles,
   --       and, in the case of some of the functions, for parameter values with
   --       sufficiently large magnitudes.
   --
   --       When one parameter of a function with multiple parameters represents a
   --       pole and another is outside the function's domain, the latter takes
   --       precedence.

-------------------------------------------------------------------------
-- TYPES
-------------------------------------------------------------------------

   Type Exception = (Not_Exception,
                     Argument_Error,
                     Constraint_Error,
                     Other_Error);

-------------------------------------------------------------------------
-- CONSTANTS
-------------------------------------------------------------------------

   constant Pi : Real := 3.14159265358979323846264338327 9;
   constant e  : Real := 2.71828182845904523536028747135 2;

-------------------------------------------------------------------------
-- FUNCTIONS
-------------------------------------------------------------------------

   -- The function Last_Exception returns the exception of the last
   -- math lib function call.
   --
   -- In case of Not_Exception the value returned by the last math
   -- lib function call was valid.
   --
   -- In case of any other value the value returned by the last math
   -- lib function call was invalid.
   --
   -- The function Last_Exception is allowed from UCL only.

  1:   function Last_Exception : Exception;

   -- In case of an exception in a math lib function the return value
   -- of the math lib function is set to 0.0!
   -- This means, only if the return value is equal to zero, you need
   -- to check the previous function call.

  2:   function Sqrt    (X     : Real) : Real;
  3:   function Log     (X     : Real;
                         Base  : Real) : Real;
  4:   function LogLn   (X     : Real) : Real;
   --
  5:   function Exp     (X     : Real) : Real;
  6:   function Power   (Left  : Real;
                         Right : Real) : Real;
```

Copyright per DIN 34

```
   --
 7:   function Sin     (X     : Real) : Real;
 8:   function Sin2    (X     : Real;
                        Cycle : Real) : Real;
 9:   function Cos     (X     : Real) : Real;
10:   function Cos2    (X     : Real;
                        Cycle : Real) : Real;
11:   function Tan     (X     : Real) : Real;
12:   function Tan2    (X     : Real;
                        Cycle : Real) : Real;
13:   function Cot     (X     : Real) : Real;
14:   function Cot2    (X     : Real;
                        Cycle : Real) : Real;
   --
15:   function Arcsin  (X     : Real) : Real;
16:   function Arcsin2 (X     : Real;
                        Cycle : Real) : Real;
17:   function Arccos  (X     : Real) : Real;
18:   function Arccos2 (X     : Real;
                        Cycle : Real) : Real;
19:   function Arctan  (Y     : Real;
                        X     : Real  := 1.0) : Real;
20:   function Arctan2 (Y     : Real;
                        X     : Real  := 1.0;
                        Cycle : Real) :  Real;
21:   function Arccot  (X     : Real;
                        Y     : Real  := 1.0) : Real;
22:   function Arccot2 (X     : Real;
                        Y     : Real  := 1.0;
                        Cycle : Real) :   Real;
   --
23:   function Sinh    (X     : Real) : Real;
24:   function Cosh    (X     : Real) : Real;
25:   function Tanh    (X     : Real) : Real;
26:   function Coth    (X     : Real) : Real;
27:   function Arsinh  (X     : Real) : Real;
28:   function Arcosh  (X     : Real) : Real;
29:   function Artanh  (X     : Real) : Real;
30:   function Arcoth  (X     : Real) : Real;

   --
31:   function Random         : Real;    -- from 0.0 to 1.0
32:   function Random_I       : Integer; -- full range

end MATH_LIB;
```

Copyright per DIN 34

I-7 UCL Ground System library: MATH_LIB_LONG

The MATH_LIB_LONG **library** contains standard mathematical operations for long_real **type** parameter

library Id (Body_Id): 7

I-7.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  MATH_LIB_LONG UCL System Library
-- ****************************************
--
-- ABSTRACT
--     Defines Functions for basic mathematic operations
--     for Long_Real type.
--     Must be compiled for ground and with Body Id = 7
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : MATH_LIB_LONG System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/math_lib_long_.ucl#5 $"
--*---------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library MATH_LIB_LONG;

------------------------------------------------------------------------

   --    GENERAL:
   --    --------
   --    1. The functions have their usual mathematical meanings. The Log function
   --       computes the logarithm to the given base; LogLn computes the natural
   --       logarithm. When the Cycle parameter is specified, the parameter X
   --       of the forward trigonometric functions (Sin2, Cos2, Tan2, and Cot2)
   --       and the results of the inverse trigonometric functions (Arcsin2,
   --       Arccos2, Arctan2, and Arccot2) are measured in units such that a
   --       full cycle of revolution has the given value; otherwise (Sin, Cos,
   --       Tan, Cot, Arcsin, Arccos, Arctan, Arccot,Sinh, Cosh, Tanh, Coth ,
   --       Arsinh, Arcosh, Artanh and Arcoth), they are measured in radians.
   --    2. The computed results of the mathematically multivalued functions are
   --       rendered single-valued by the following conventions, which are meant to
   --       imply the principal branch:
   --           1.The results of the Sqrt and Arcosh functions and that of the
   --             exponentiation operator are nonnegative.
   --           2.The result of the Arcsin function is in the quadrant containing the
   --             point (1.0, x), where x is the value of the parameter X. This
   --             quadrant is I or IV; thus, the range of the Arcsin function is
   --             approximately -Pi/2.0 to Pi/2.0 (-Cycle/4.0 to Cycle/4.0, if the
   --             parameter Cycle is specified).
   --           3.The result of the Arccos function is in the quadrant containing the
   --             point (x, 1.0), where x is the value of the parameter X. This
   --             quadrant is I or II; thus, the Arccos function ranges from 0.0 to
   --             approximately Pi (Cycle/2.0, if the parameter Cycle is specified).
   --           4.The results of the Arctan and Arccot functions are in the quadrant
   --             containing the point (x, y), where x and y are the values of the
   --             parameters X and Y, respectively. This may be any quadrant
   --             (I through IV) when the parameter X (resp., Y) of Arctan (resp.,
   --             Arccot) is specified, but it is restricted to quadrants I and IV
   --             (resp., I and II) when that parameter is omitted. Thus, the range
   --             when that parameter is specified is approximately -Pi to Pi
   --             (-Cycle/2.0 to Cycle/2.0, if the parameter Cycle is specified); when
   --             omitted, the range of Arctan (resp., Arccot) is that of
   --             Arcsin (resp., Arccos), as given above. When the point (x, y) lies
   --             on the negative x-axis, the result approximates Pi (resp., -Pi) when
   --             the sign of the parameter Y is positive (resp., negative)
   --
   --
   --       (In the case of the inverse trigonometric functions, in which a result
   --       lying on or near one of the axes may not be exactly representable, the
   --       approximation inherent in computing the result may place it
   --       in an adjacent quadrant, close to but on the wrong side of the axis.)
   --
   --    3. The random functions returns a value in integer range (Random_I)
   --       or in range 0.0 .. 1.0 (Random).
   --
   --    4. The function Last_Exception returns the exception of the last math lib
   --       function call. In case of Not_Exception the value returned by the last math
   --       lib function call was valid. In case of any other value the value returned
   --       by the last math lib function call was invalid.
   --
   --     Dynamic Semantics:
   --     ------------------
```

Copyright per DIN 34

```
   --     2.The exception Argument_Error (function Last_Exception) is raised,
   --       signaling a parameter value outside the domain of the corresponding
   --       mathematical function, in the following cases:
   --           1.by any forward or inverse trigonometric function with specified
   --             cycle, when the value of the parameter Cycle is zero or negative;
   --           2.by the Log function with specified base, when the value of the
   --             parameter Base is zero, one, or negative;
   --           3.by the Sqrt and Log functions, when the value of the parameter
   --             X is negative;
   --           4.by the exponentiation operator, when the value of the left
   --             operand is negative or when both operands have the value zero;
   --           5.by the Arcsin, Arccos, and Artanh functions, when the absolute
   --             value of the parameter X exceeds one;
   --           6.by the Arctan and Arccot functions, when the parameters X and Y
   --             both have the value zero;
   --           7.by the Arcosh function, when the value of the parameter X is
   --             less than one; and
   --           8.by the Arcoth function, when the absolute value of the parameter
   --             X is less than one.
   --
   --     1.The exception Constraint_Error (function Last_Exception) is raised,
   --       signaling a pole of the mathematical function (analogous to dividing
   --       by zero), in the following cases:
   --           1.by the Log, Cot, and Coth functions, when the value of the parameter
   --             X is zero;
   --           2.by the exponentiation operator, when the value of the left operand
   --             is zero and the value of the exponent is negative;
   --           3.by the Tan function with specified cycle, when the value of the
   --             parameter X is an odd multiple of the quarter cycle;
   --           4.by the Cot function with specified cycle, when the value of the
   --             parameter X is zero or a multiple of the half cycle; and
   --           5.by the Artanh and Arcoth functions, when the absolute value
   --             of the parameter X is one.
   --
   --       Constraint_Error can also be raised when a finite result overflows;
   --       this may occur for parameter values sufficiently near poles,
   --       and, in the case of some of the functions, for parameter values with
   --       sufficiently large magnitudes.
   --
   --       When one parameter of a function with multiple parameters represents a
   --       pole and another is outside the function's domain, the latter takes
   --       precedence.

-------------------------------------------------------------------------
-- TYPES
-------------------------------------------------------------------------

   Type Exception = (Not_Exception,
                     Argument_Error,
                     Constraint_Error,
                     Other_Error);

-------------------------------------------------------------------------
-- CONSTANTS
-------------------------------------------------------------------------

   constant Pi : Long_Real := 3.14159265358979323846264338327 9;
   constant e  : Long_Real := 2.71828182845904523536028747135 2;

-------------------------------------------------------------------------
-- FUNCTIONS
-------------------------------------------------------------------------

   -- The function Last_Exception returns the exception of the last
   -- math lib function call.
   --
   -- In case of Not_Exception the value returned by the last math
   -- lib function call was valid.
   --
   -- In case of any other value the value returned by the last math
   -- lib function call was invalid.
   --
   -- The function Last_Exception is allowed from UCL only.

  1:   function Last_Exception : Exception;

   -- In case of an exception in a math lib function the return value
   -- of the math lib function is set to 0.0!
   -- This means, only if the return value is equal to zero, you need
   -- to check the previous function call.

  2:   function Sqrt    (X     : Long_Real) : Long_Real;
  3:   function Log     (X     : Long_Real;
                         Base  : Long_Real) : Long_Real;
  4:   function LogLn   (X     : Long_Real) : Long_Real;
   --
  5:   function Exp     (X     : Long_Real) : Long_Real;
  6:   function Power   (Left  : Long_Real;
                         Right : Long_Real) : Long_Real;
   --
  7:   function Sin     (X     : Long_Real) : Long_Real;
```

Copyright per DIN 34

```
  8:   function Sin2    (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
  9:   function Cos     (X     : Long_Real) : Long_Real;
 10:   function Cos2    (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
 11:   function Tan     (X     : Long_Real) : Long_Real;
 12:   function Tan2    (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
 13:   function Cot     (X     : Long_Real) : Long_Real;
 14:   function Cot2    (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
   --
 15:   function Arcsin  (X     : Long_Real) : Long_Real;
 16:   function Arcsin2 (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
 17:   function Arccos  (X     : Long_Real) : Long_Real;
 18:   function Arccos2 (X     : Long_Real;
                         Cycle : Long_Real) : Long_Real;
 19:   function Arctan  (Y     : Long_Real;
                         X     : Long_Real  := 1.0) : Long_Real;
 20:   function Arctan2 (Y     : Long_Real;
                         X     : Long_Real  := 1.0;
                         Cycle : Long_Real) :  Long_Real;
 21:   function Arccot  (X     : Long_Real;
                         Y     : Long_Real  := 1.0) : Long_Real;
 22:   function Arccot2 (X     : Long_Real;
                         Y     : Long_Real  := 1.0;
                         Cycle : Long_Real) :   Long_Real;
   --
 23:   function Sinh    (X     : Long_Real) : Long_Real;
 24:   function Cosh    (X     : Long_Real) : Long_Real;
 25:   function Tanh    (X     : Long_Real) : Long_Real;
 26:   function Coth    (X     : Long_Real) : Long_Real;
 27:   function Arsinh  (X     : Long_Real) : Long_Real;
 28:   function Arcosh  (X     : Long_Real) : Long_Real;
 29:   function Artanh  (X     : Long_Real) : Long_Real;
 30:   function Arcoth  (X     : Long_Real) : Long_Real;

   --
 31:   function Random         : Long_Real;    -- from 0.0 to 1.0
 32:   function Random_I       : Integer;      -- full range

end MATH_LIB_LONG;
```

Copyright per DIN 34

I-8 UCL Ground System library: MESSAGE_LIBRARY

The MESSAGE_LIBRARY **library** contains procedures to access system messages.

library Id (Body_Id): 19

I-8.1 UCL System library Specification

```
--
-- ****************************************
--  MESSAGE_LIBRARY UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to access system messages.
--     Must be compiled for ground and with Body Id = 19
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  :  MESSAGE_LIBRARY System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/message_library_.ucl#5 $"
--*-------------------------------------------------------------------
---
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
--
------------------------------------------------------------------------

library MESSAGE_LIBRARY;

  ------------------------------------------------------------------------
  -- IMPORTS
  ------------------------------------------------------------------------

  import GROUND_COMMON;

  ------------------------------------------------------------------------
  -- TYPES
  ------------------------------------------------------------------------

  type MESSAGE_CLASSES = (FATAL, SEVERE, ORDINARY, ADVISORY);
  type MESSAGE_CLASS_SET = set of MESSAGE_CLASSES;

  ------------------------------------------------------------------------
  -- CONSTANTS
  ------------------------------------------------------------------------


  constant DEFAULT_TIMEOUT:  DURATION := 5.0 [s];
  constant ALL_MESSAGES:     MESSAGE_CLASS_SET
                          := MESSAGE_CLASS_SET{FATAL, SEVERE, ORDINARY, ADVISORY};


  ------------------------------------------------------------------------
  -- PROCEDURES
  ------------------------------------------------------------------------

  1:  procedure CREATE_MESSAGE_BUFFER
        (in  BUFFER_SIZE           : UNSIGNED_INTEGER  := 500;
         in  DELETE_NEWEST_MESSAGE : BOOLEAN           := TRUE;
         in  MESSAGE_CLASSES       : MESSAGE_CLASS_SET := ALL_MESSAGES;
         out STATUS                : UCL_RETURN);

  -- Description:
  --   Create one global buffer for all system messages with matching message
  --   class. System messages are all messages, except messages generated by
  --   user by ground_library calls write_message_to_user or log.
  -- Parameters:
  --   IN:
  --     BUFFER_SIZE :           maximal size of global buffer
  --     DELETE_NEWEST_MESSAGE : In case of buffer overflow, delete the newest
  --                             message, let the full buffer unchanged.
  --                             If this value is set to FALSE, the oldest message
  --                             will be overwritten by the newest message.
  --     MESSAGE_CLASSES :       set of message classes, which shall insert
  --                             into buffer
  --   OUT:
  --     STATUS returns the UCL Return Code values :
  --       OK
  --       ITEM_IS_ENABLED    buffer already exists
  --       NOT_OK


  2:  procedure CLOSE_MESSAGE_BUFFER
        (out STATUS: UCL_RETURN);
```

Copyright per DIN 34

```
   -- Description:
   --   Clear the global buffer and close it.
   -- Parameters:
   --   OUT:
   --     STATUS returns the UCL Return Code values :
   --       OK
   --       NOT_OK


   3:  procedure GET_NEXT_MESSAGE
         (in  TIMEOUT:        DURATION    := DEFAULT_TIMEOUT;
          out TEXT:           STRING;
          out SUPPLEMENT:     STRING;
          out MESSAGE_GROUP:  MSG_GROUP;
          out MESSAGE_TYPE:   MSG_TYPE;
          out MESSAGE_CLASS:  MESSAGE_CLASSES;
          out MESSAGE_NUMBER: UNSIGNED_INTEGER;
          out NODE:           STRING;
          out TIME_TAG_LT:    TIME;
          out TIME_TAG_SMT:   TIME;
          out AP:             PATHNAME;
          out AP_ID:          UNSIGNED_INTEGER;
          out STATUS:         UCL_RETURN);

   -- Description:
   --   Get the next exception and remove it from buffer.
   --   If no exception is available, wait until timeout or occurrence
   --   of an exception.
   -- Parameters:
   --   IN:
   --     TIMEOUT :  maximal time for waiting on next exception
   --   OUT:
   --     MESSAGE:        content of message
   --     SUPPLEMENT:     additional information
   --     MESSAGE_GROUP:  group of message as string
   --     MESSAGE_TYPE:   type of message as string
   --     MESSAGE_CLASS:  message class
   --     MESSAGE_NUMBER: internal number of message in message buffer
   --     NODE:           application name, which sent this message as string
   --     TIME_TAG_LT:    time tag of message in local time
   --     TIME_TAG_SMT:   time tag of message in simulated mission time
   --     AP:             if this information is available, which AP caused
   --                     this message, \\ otherwise
   --     AP_ID:          if ap = \\ the ap_id is set to 0,
   --                     otherwise the ap_id
   --     STATUS returns the UCL Return Code values :
   --       OK
   --       TIMEOUT       no message in buffer during wait
   --       NOT_OK        buffer not active

   4:  function MESSAGES_IN_BUFFER : UNSIGNED_INTEGER;

   -- Description:
   --   Returns the number of messages in message buffer.
   --   Returns 0 if buffer is empty or buffer not active.

end MESSAGE_LIBRARY;
```

Copyright per DIN 34

I-9 UCL Ground System library: TC_CONSTRUCTION

The TC CONSTRUCTION **library** contains operations to construct a Telecommand online in UCL.

library Id (Body_Id): 9

I-9.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  TC_CONSTRUCTION UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to construct Telecommands.
--
--     Must be compiled with Body Id = 9
--
--     recommended Nickname: "TC_CONSTRUCTION"
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : TC_CONSTRUCTION System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/tc_construction_.ucl#5 $"
--*----------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library TC_CONSTRUCTION;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

   import GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES and CONSTANTS
------------------------------------------------------------------------

  constant DEFAULT_ISSUE_TIMEOUT: DURATION := 0.5 [s];

  type HANDLE     = INTEGER;
  -- Remark:
  --   The number of TCs to be constructed is limited to 22 and maintained by
  --   TES internaly. An error will be returned if this limit is exceeded.

  type VERSION_RANGE   = INTEGER (0 .. 7);

  type TYPE_RANGE      = INTEGER (0 .. 1);

  type APID_RANGE      = INTEGER (0 .. 2**11 - 1);

  type SEG_FLAG_RANGE  = INTEGER (0 .. 3);

  type SEQ_COUNT_RANGE = INTEGER (0 .. 2**14 - 1);

  -- size declarations
  constant PRIMARY_HEADER_SIZE: INTEGER := 6;
  constant MAX_DATA_SIZE: INTEGER := 2**12 - PRIMARY_HEADER_SIZE;

  type ADDRESS            = INTEGER (0 .. MAX_DATA_SIZE - 1);
  -- Remark:
  --   length of the data field of a TC packet (excluding primary header)

  type OFFSET_TYPE        = (BIT_OFFSET, BYTE_OFFSET);
  -- Remark:
  --   used to set new data: the data may be replaced at a bit or a byte position

  type OFFSET             = INTEGER (1 .. MAX_DATA_SIZE * 8);
  -- Remark:
  --   maximum bit offset in a TC data field
  --   note: offset 1 means: first bit/byte in the data field BEHIND the primary header

  type BIT                = INTEGER (0 .. 1);

  type CCSDS_HEADER_RANGE = INTEGER (0 .. 1);

  type CONSTRUCT_ITEM_NAME =
    PATHNAME (EGSE_BINARY_PACKET,
              EGSE_PREDEFINED_TC,
              PUS_TC);

  type TC_PACKET_TYPE = (PUS_TC, EGSE_PREDEFINED_TC, EGSE_BINARY_PACKET);
```

Copyright per DIN 34

```
  type CHECKSUM_TYPE  = (NONE, CRC16, ADD_WO_CARRY);
  -- Remark:
  --   Two different checksums are available:
  --     o CRC16 polynomial: x**16 + x**15 + x**2 + 1 (CRC16)
  --   or
  --     o 16 bit add without carry (as used by COL)
  --
  --   NONE is used to create/send a TC without checksum

  constant DEFAULT_CHECKSUM: CHECKSUM_TYPE := CRC16;

  type BYTE_ARRAY  = STRING (MAX_DATA_SIZE) of BYTE;
  type CHAR_ARRAY  = STRING (MAX_DATA_SIZE) of CHARACTER;

  type SAS_NAME    = STRING(20);


-------------------------------------------------------------------------
-- OPERATIONS
-------------------------------------------------------------------------

  -------------------------------------------------------------------------
  --   Create new TC
  -------------------------------------------------------------------------

  1:  procedure CONSTRUCT_FROM_TC
        (in  TC:        CONSTRUCT_ITEM_NAME ();
         out TC_HANDLE: HANDLE;
         out STATUS:    UCL_RETURN);

  -- Description:
  --   This procedure will fetch an existing TC declaration from the database,
  --   which can be modified using the SET-/APPEND operations from this library.
  --
  --   Note (1): The CONSTRUCT procedures will allocate new memory.
  --         Therefore, the maximum number of handles, i.e. the number of TC,
  --         is limited. The allocated memory should be freed, if a TC is not
  --         used anymore.
  --  Note (2): The TC data as well as the primary header fields are fetched
  --         from the database.
  --         Parameters to the TC defined in the database will not be taken
  --         into account.
  --         Try to set any header field will result in an error, if the
  --         TC has been constructed from an EGSE_BINARY_PACKET
  -- Parameters:
  --   IN:
  --     TC        : pathname of telecommand
  --   OUT:
  --     TC_HANDLE : handle to access this TC
  --     STATUS    : UCL return status
  --                   OK
  --                   NO_FREE_HANDLE
  --                   RUNTIME_ERROR

  2:  procedure CONSTRUCT_FROM_HANDLE
        (in  FROM_TC_HANDLE:   HANDLE;
         out TC_HANDLE:        HANDLE;
         out STATUS:           UCL_RETURN);

  -- Description:
  --   This procedure will use the TC definition from an already defined
  --   handle.
  -- Parameters:
  --   IN:
  --     FROM_TC_HANDLE : handle to TC
  --   OUT:
  --     TC_HANDLE      : handle to access this TC
  --     STATUS         : UCL return status
  --                   OK
  --                   NO_FREE_HANDLE
  --                   INVALID_TC_HANDLE
  --                   RUNTIME_ERROR

  3:  procedure CONSTRUCT
        (in  TC_TYPE:           TC_PACKET_TYPE := PUS_TC;
         out TC_HANDLE:         HANDLE;
         out STATUS:            UCL_RETURN);

  -- Description:
  --   This operation can be used to define a new TC. The entire
  --   TC will be completely cleared, i.e. all data including the
  --   primary header has to be set manually using the SET-/APPEND
  --   procedures.
  -- Parameters:
  --   IN:
  --     TC_TYPE   : type of TC (see TC_PACKET_TYPE)
  --   OUT:
  --     TC_HANDLE : handle to access this TC
```

Copyright per DIN 34

```
--     STATUS    : UCL return status
--                 OK
--                 NO_FREE_HANDLE
--                 RUNTIME_ERROR

4:  procedure FREE
      (in  TC_HANDLE:         HANDLE;
       out STATUS:            UCL_RETURN);

-- Description:
--   This operation will deallocate the memory used to construct the
--   TC specified by TC_HANDLE. Any access to this TC will result in
--   an error, unless it is reused by the next call of CONSTRUCT,
--   CONSTRUCT_FROM_TC or CONSTRUCT_FROM_HANDLE.
-- Parameters:
--   IN:
--     TC_HANDLE : handle to deallocate this TC
--   OUT:
--     STATUS    : UCL return status
--                 OK
--                 INVALID_TC_HANDLE
--                 RUNTIME_ERROR

5:  procedure SET_VERSION_NUMBER
      (in  TC_HANDLE:        HANDLE;
       in  VERSION_NUMBER:   VERSION_RANGE;
       out STATUS:           UCL_RETURN);

-- Description:
--   Assign new values to an already created TC
--   This procedure will assign a new version number within the CCSDS
--   Primary Header of the specified TC.
--   Default value for CONSTRUCT: 0
-- Parameters:
--   IN:
--     TC_HANDLE      : handle to access this TC
--     VERSION_NUMBER : number of version
--   OUT:
--     STATUS         : UCL return status
--                 OK
--                 INVALID_TC_HANDLE
--                 RUNTIME_ERROR

6:  procedure SET_TYPE
      (in  TC_HANDLE:     HANDLE;
       in  TYPE_NUMBER:   TYPE_RANGE;
       out STATUS:        UCL_RETURN);

-- Description:
--   This procedure will assign a new type within the CCSDS
--   Primary Header of the specified TC.
--   Default value for CONSTRUCT: 1
-- Parameters:
--   IN:
--     TC_HANDLE   : handle to access this TC
--     TYPE_NUMBER : number of type
--   OUT:
--     STATUS      : UCL return status
--                   OK
--                   INVALID_TC_HANDLE
--                   RUNTIME_ERROR

7:  procedure SET_2ND_HEADER_FLAG
      (in  TC_HANDLE:    HANDLE;
       in  CCSDS_VALUE:  CCSDS_HEADER_RANGE;
       out STATUS:       UCL_RETURN);

-- Description:
--   This procedure will assign a new CCSDS 2nd Header Bit within
--   the Primary Header of the specified TC.
--   Default value for CONSTRUCT: 1 (with secondary header)
-- Parameters:
--   IN:
--     TC_HANDLE   : handle to access this TC
--     CCSDS_VALUE : number of type
--   OUT:
--     STATUS      : UCL return status
--                   OK
--                   INVALID_TC_HANDLE
--                   RUNTIME_ERROR

8:  procedure SET_APID
      (in  TC_HANDLE:    HANDLE;
       in  APID:         APID_RANGE;
       out STATUS:       UCL_RETURN);

-- Description:
--   This procedure will assign a new CCSDS application process id within
--   the Primary Header of the specified TC.
--   Default value for CONSTRUCT: 0
-- Parameters:
```

Copyright per DIN 34

```
--   IN:
--     TC_HANDLE  : handle to access this TC
--     APID       : number of type
--   OUT:
--     STATUS     : UCL return status

 9:  procedure SET_SEGMENT_FLAGS
       (in  TC_HANDLE:    HANDLE;
        in  SEG_FLAGS:    SEG_FLAG_RANGE;
        out STATUS:       UCL_RETURN);

-- Description:
--   This procedure will assign new segment flags within the Primary
--   Header of the specified TC.
--   Default value for CONSTRUCT: 3
-- Parameters:
--   IN:
--     TC_HANDLE : handle to access this TC
--     SEG_FLAGS : segment flags
--   OUT:
--     STATUS      : UCL return status
--                   OK
--                   INVALID_TC_HANDLE
--                   RUNTIME_ERROR

10:  procedure SET_SEQUENCE_COUNT
       (in  TC_HANDLE:    HANDLE;
        in  SEQ_COUNT:    SEQ_COUNT_RANGE;
        out STATUS:       UCL_RETURN);

-- Description:
--   This procedure will assign a new source sequence counter within
--   the Primary Header of the specified TC.
--   Default value for CONSTRUCT: 0
--
--   Note: the sequence counter will automatically be set by procedure
--         SEND, unless it has already been set via SET_SEQUENCE_COUNT
--
-- Parameters:
--   IN:
--     TC_HANDLE : handle to access this TC
--     SEQ_COUNT : sequence count
--   OUT:
--     STATUS      : UCL return status
--                   OK
--                   INVALID_TC_HANDLE
--                   RUNTIME_ERROR

11:  procedure SET_PACKET_LENGTH
       (in  TC_HANDLE:     HANDLE;
        in  PACKET_LENGTH: ADDRESS;
        out STATUS:        UCL_RETURN);

-- Description:
--   This procedure will set a new packet length within the
--   Primary Header of the specified TC.
--   Default value for CONSTRUCT: 0
--
--   Note: normally the length in a CCSDS packet denotes the
--         length of the data part plus 1, i.e. 0 means length = 1,
--         1 means length = 2 etc. Unfortunately, in case of length = 0,
--         the append operation would add the data behind the first data byte
--         then. Therefore, a special handling has been implemented here
--         to allow adding data just behind the primary header.
--
-- Parameters:
--   IN:
--     TC_HANDLE     : handle to access this TC
--     PACKET_LENGTH : packet length
--   OUT:
--     STATUS        : UCL return status
--                     OK
--                     INVALID_TC_HANDLE
--                     RUNTIME_ERROR

12:  procedure SET_BYTE_STRING
       (in  TC_HANDLE:   HANDLE;
        in  AT_POS:      OFFSET;
        in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
        in  VALUE:       BYTE_ARRAY;
        out STATUS:      UCL_RETURN);

-- Description:
--   This procedures will replace a value at the provided
--   address within a TC. The address specifies the BIT- or BYTE-offset
--   within the data field of the TC.
--
--   Note: the address of the first bit/byte is 1. The address range,
--         thus, is: 1 .. 32720 (for BIT_OFFSET) and 1 .. 4090 (for BYTE_OFFSET).
--
-- Parameters:
```

Copyright per DIN 34

```
--   IN:
--     TC_HANDLE:   handle to access this TC
--     AT_POS:      offset
--     OFFSET_KIND: offset kind
--     VALUE:       value to be set
--   OUT:
--     STATUS     : UCL return status
--                   OK
--                   INVALID_TC_HANDLE
--                   RUNTIME_ERROR

13:  procedure SET_CHAR_STRING
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       CHAR_ARRAY;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   see above (SET_BYTE_STRING)

14:  procedure SET_HEX_STRING
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       CHAR_ARRAY;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   This procedure will convert a hexadecimal string to its
 --   binary representation.
 --
 --   Example:  "FF" will be converted to the value 255
 --           "FF10" will be converted to 255 and 16
 --
 --   Note: the length of the string must be a multiple of 2, since
 --         always two characters will be converted to one single byte
 --
 -- Parameters:
 --   IN:
 --     TC_HANDLE:   handle to access this TC
 --     AT_POS:      offset
 --     OFFSET_KIND: offset kind
 --     VALUE:       value to be set
 --                  Hex-String e.g. "00FFAF6799"
 --   OUT:
 --     STATUS      : UCL return status
 --                   OK
 --                   INVALID_TC_HANDLE
 --                   RUNTIME_ERROR
 --                   INVALID_SIZE      : size of the VALUE string is invalid
 --                   INVALID_PARAMETER : VALUE string contains any character,
 --                                       which cannot be converted

15:  procedure SET_BIT
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       BIT;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   see above (SET_BYTE_STRING)

16:  procedure SET_BYTE
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       BYTE;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   see above (SET_BYTE_STRING)

17:  procedure SET_INTEGER
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       INTEGER;
       out STATUS:      UCL_RETURN);

 -- Description:
 --   see above (SET_BYTE_STRING)

18:  procedure SET_UNSIGNED_INTEGER
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       UNSIGNED_INTEGER;
       out STATUS:      UCL_RETURN);
```

Copyright per DIN 34

```
-- Description:
--   see above (SET_BYTE_STRING)

19:  procedure SET_REAL
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       REAL;
       out STATUS:      UCL_RETURN);

-- Description:
--   see above (SET_BYTE_STRING)

20:  procedure SET_LONG_REAL
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       LONG_REAL;
       out STATUS:      UCL_RETURN);

-- Description:
--   see above (SET_BYTE_STRING)

21:  procedure SET_TIME
      (in  TC_HANDLE:   HANDLE;
       in  AT_POS:      OFFSET;
       in  OFFSET_KIND: OFFSET_TYPE := BIT_OFFSET;
       in  VALUE:       TIME;
       out STATUS:      UCL_RETURN);

-- Description:
--   see above (SET_BYTE_STRING)

22:  procedure APPEND_FROM_TC
      (in  TO_TC_HANDLE:  HANDLE;
       in  FROM_TC:       CONSTRUCT_ITEM_NAME ();
       in  CHECKSUM:      CHECKSUM_TYPE := DEFAULT_CHECKSUM;
       out STATUS:        UCL_RETURN);

-- Description:
--   This procedures will append a TC to the application
--   data field of the specified TC. The length of the entire TC will
--   be automatically recalculated and the length field of the primary
--   header will be modified accordingly.
--   The PEC, i.e. Packet Error Control, will be recalculated, if the
--   TC is send (SEND_TC) or appended to another TC. In the latter case,
--   only the PEC of the inserted TC will be recalculated.
--
--   Note: PEC calculation can be inhibited using the value NONE for the
--         parameter CHECKSUM of the APPEND/SEND_TC operations.
--
-- Parameters:
--   IN:
--     TO_TC_HANDLE:  handle to access this TC
--     FROM_TC:       telecommand, which is append
--     CHECKSUM:      type of checksum
--   OUT:
--     STATUS       : UCL return status
--                    OK
--                    INVALID_TC_HANDLE
--                    TC_NOT_YET_DEFINED
--                    INVALID_SIZE
--                    RUNTIME_ERROR

23:  procedure APPEND_FROM_HANDLE
      (in  TO_TC_HANDLE:  HANDLE;
       in  FROM_TC:       HANDLE;
       in  CHECKSUM:      CHECKSUM_TYPE := DEFAULT_CHECKSUM;
       out STATUS:        UCL_RETURN);

-- Description:
--   see above (APPEND_FROM_TC)

24:  procedure APPEND_BYTE_STRING
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        BYTE_ARRAY;
       out STATUS:       UCL_RETURN);

-- Description:
--   This procedures will append a value in the application
--   data field of the specified TC. The length of the entire TC will
--   be automatically recalculated and the length field of the primary
--   header will be modified accordingly.
-- Parameters:
--   IN:
--     TO_TC_HANDLE:  handle to access this TC
--     VALUE:         value, which is append
--   OUT:
--     STATUS       : UCL return status
--                    OK
--                    INVALID_TC_HANDLE
```

Copyright per DIN 34

```
--                  TC_NOT_YET_DEFINED
--                  INVALID_SIZE
--                  RUNTIME_ERROR


25:  procedure APPEND_CHAR_STRING
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        CHAR_ARRAY;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

26:  procedure APPEND_HEX_STRING
   (in  TO_TC_HANDLE: HANDLE;
    in  VALUE:        CHAR_ARRAY;
    out STATUS:       UCL_RETURN);

 -- Description:
 --   This procedure will append a converted hexadecimal string as its
 --   binary representation.
 --
 --   Example: "FF" will be converted to the value 255
 --            "FF10" will be converted to 255 and 16
 --
 --   Note: the length of the string must be a multiple of 2, since
 --         always two characters will be converted to one single byte
 --
 -- Parameters:
 --   IN:
 --     TC_HANDLE:   handle to access this TC
 --     VALUE:       value to be set
 --                  Hex-String e.g. "00FFAF6799"
 --   OUT:
 --     STATUS      : UCL return status
 --                    OK
 --                    INVALID_TC_HANDLE
 --                    RUNTIME_ERROR
 --                    INVALID_SIZE      : size of the VALUE string is invalid
 --                    INVALID_PARAMETER : VALUE string contains any character,
 --                                        which cannot be converted

27:  procedure APPEND_BIT
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        BIT;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

28:  procedure APPEND_BYTE
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        BYTE;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

29:  procedure APPEND_INTEGER
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        INTEGER;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

30:  procedure APPEND_UNSIGNED_INTEGER
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        UNSIGNED_INTEGER;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

31:  procedure APPEND_REAL
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        REAL;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

32:  procedure APPEND_LONG_REAL
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        LONG_REAL;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

33:  procedure APPEND_TIME
```

Copyright per DIN 34

```
      (in  TO_TC_HANDLE: HANDLE;
       in  VALUE:        TIME;
       out STATUS:       UCL_RETURN);

 -- Description:
 --   see above (APPEND_BYTE_STRING)

34:  guarded procedure SEND_TC
      (in  TC:                     HANDLE;
       in  PRIO:                   PRIORITY := GROUND_COMMON.LOW;
       in  AT_TIME:                TIME := ~:~;
       in  BASE:                   TIME_BASE := LOCAL_TIME;
       in  ONBOARD_EXECUTION_TIME: TIME := ~:~;
       in  TIMEOUT:                DURATION := DEFAULT_ISSUE_TIMEOUT;
       in  TO_SAS:                 SAS_NAME;
       in  CHECKSUM:               CHECKSUM_TYPE := DEFAULT_CHECKSUM;
       out SAS_ACK_CODE:           INTEGER;
       out SAS_ACK_TIME:           TIME;
       out CCSDS_SEQUENCE_COUNT:   INTEGER;
       out STATUS:                 UCL_RETURN);

 -- Description:
 --   This operations will send a constructed TC to the specified
 --   SAS. Note that the information assigned to a TC via the database,
 --   like physical address, will be used in case the TC has been defined
 --   using CONSTRUCT_FROM_TC. Default values will be used in any other case.
 -- Parameters:
 --   IN:
 --     TC :      the handle, pointed to enditem, which is to be issued.
 --     PRIO    : the priority to be used for issuing the item
 --               (GROUND_COMMON.LOW or GROUND_COMMON.HIGH)
 --     AT_TIME : The time to issue the stimulus from application (SAS)
 --               Default: no time.
 --
 --               The purpose of the AT_TIME parameter shall be the same
 --               for all GDU alternatives:
 --
 --               For MDB enditems of type EGSE_BINARY_PACKET and
 --               EGSE_PREDEFINED_TC,  the time value is written as the
 --               TIME_TAG into the GDU generated by the command and sent to
 --               the SAS. The SAS has to extract the TIME_TAG from the
 --               GDU and execute the stimulus at the given time.
 --
 --     BASE    : specifies if AT_TIME is local time or SMT
 --
 --     ONBOARD_EXECUTION_TIME : The time to execute the telecommand onboard
 --               Default:  no time.
 --
 --               For MDB Enditems of type EGSE_PREDEFINED_TC, the
 --               ONBOARD_EXECUTION _TIME value will be written to the TIME
 --               field of the secondary header of the corresponding CCSDS
 --               packet. In this case, the TIME_ID field of the secondary
 --               header shall be set to '10'B, which means: a time tag is
 --               present. No interpretation of TIME_BASE will be done by CGS
 --               in this case. The TIME_TAG of the GDU header generated by
 --               the command will be independent from this.
 --
 --               If no ONBOARD_EXECUTION _TIME parameter is given, i.e. the
 --               default value is applied , the TIME_ID and TIME field shall
 --               be set to '10'B (= time tagged command: the execution of the
 --               command is scheduled by the onboard system to the specified
 --               time)
 --
 --               Remark: If the TIME_ID is of value '00'B (UNDEFINED) or
 --               '11'B (NO_TIME_TAG), the TIME field in the secondary header
 --               shall be set to zero via the database default entry, in case
 --               no ONBOARD_EXECUTION_TIME parameter is given.
 --
 --     TIMEOUT : the maximum time to wait for acknowledgement of this
 --               stimulus or telecommand
 --     TO_SAS :  SAS name, to which the telecommand is to be issued
 --     CHECKSUM : the type of checksum
 --
 --        Note: PEC calculation can be inhibited using the value NONE for the
 --              parameter CHECKSUM.
 --
 --   OUT:
 --     SAS_ACK_CODE : the return code received from SAS
 --                    range (0, [ 1000 .. 2**16 - 1 ])
 --     SAS_ACK_TIME : the time received from SAS
 --     CCSDS_SEQUENCE_COUNT : the value used as sequence counter in a issued
 --                     CCSDS Packet (single CCSDS packats only)
 --     STATUS : UCL return status
 --                   OK
 --                   INVALID_TESTNODE_MODE
 --                   NO_GDU_SERVICE
 --                   APPLICATION_NACK
 --                   TIMEOUT
 --                   ITEM_IS_DISABLED
 --                   APPLICATION_NOT_READY
 --                   INVALID_PARAMETER
```

Copyright per DIN 34

```
  --                  PRECONDITION_CHECK_FAILED
  --                  AUTHORIZATION_FAILED

END TC_CONSTRUCTION;
```

Copyright per DIN 34

I-10 UCL Ground System library: RAW_DATA_library

The RAW_DATA **library** contains operations to access a raw data packet

library Id (Body_Id): 12

I-10.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  RAW_DATA_LIBRARY UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to retrieve data from raw TM packets
--
--     Must be compiled for ground and with Body Id = 12
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : RAW_DATA_LIBRARY System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/raw_data_library_.ucl#5 $"
--*----------------------------------------------------------
---
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library RAW_DATA_LIBRARY;

------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------

   constant Max_Length: Integer := 4096+6;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type T_Packet      = String(Max_Length) of Byte;


   type T_Position    = Unsigned_Integer(1 .. Max_Length*8);

   type T_Bit         = Unsigned_Integer (0 .. 1);
   type T_Byte        = Unsigned_Integer (0 .. 255);
   type T_Word        = Unsigned_Integer;

   type T_Word_Size   = Unsigned_Integer (0 .. 32);

   type T_Raw_Status = (R_Success, R_Invalid_Size, R_Range_Error,
                        R_Other_Error);


   type T_Byte_Length =  Unsigned_Integer (0 .. Max_Length);


------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  1:   procedure Get_Raw_Status_Image
         (in  Status :   T_Raw_Status;
          out Image  :   string);

   -- Description:
   --   String-Conversion of T_Raw_Status


  2:   procedure Get_Raw_Bit
         (in  Packet:           T_Packet;
          in  Bit_Position:     T_Position;
          out Value:            T_Bit;
          out Status:           T_Raw_Status);

   -- Description:
   --   Retrieves one bit located at Bit_Position (starting at
   --   first Bit of primary header (Bit_Position: 1)) from Packet
   -- Parameters:
   --   OUT:
   --     Status:
   --       R_Success      retrieval was successful
   --       R_Range_Error  range violation
   --                      ( Bit_Position  > packet bit length)
```

Copyright per DIN 34

```
3:   procedure Get_Raw_Byte
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       out Value:            T_Byte;
       out Status:           T_Raw_Status);

 -- Description:
 --   Retrieves one byte located at Bit_Position (starting at
 --   first Bit of primary header (Bit_Position: 1)) from Packet
 -- Parameters:
 --   OUT:
 --     Status:
 --       R_Success      retrieval was successful
 --       R_Range_Error  range violation
 --                      (Bit_Position+8-1 > packet bit length)


4:   procedure Get_Raw_Halfword
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       out Value:            T_Word;
       out Status:           T_Raw_Status);

 -- Description:
 --   Retrieves a halfword (16 bit) located at Bit_Position (starting
 --   at first Bit of primary header (Bit_Position: 1)) from Packet
 -- Parameters:
 --   OUT:
 --     Status:
 --       R_Success      retrieval was successful
 --       R_Range_Error  range violation
 --                      (Bit_Position+16-1 > packet bit length)

5:   procedure Get_Raw_Word
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       out Value:            T_Word;
       out Status:           T_Raw_Status);

 -- Description:
 --   Retrieves one word (32 bit) located at Bit_Position (starting
 --   at first Bit of primary header (Bit_Position: 1)) from Packet
 -- Parameters:
 --   OUT:
 --     Status:
 --       R_Success      retrieval was successful
 --       R_Range_Error  range violation
 --                      (Bit_Position+32-1 > packet bit length)


6:   procedure Get_Raw_Integer
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       in  Bit_Length:       T_Word_Size;
       out Value:            Integer;
       out Status:           T_Raw_Status);
7:   procedure Get_Raw_Unsigned_Integer
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       in  Bit_Length:       T_Word_Size;
       out Value:            Unsigned_Integer;
       out Status:           T_Raw_Status);

 -- Description (Get_Raw_Integer/Get_Raw_Unsigned_Integer):
 --   Retrieves Bit_Length bits located at Bit_Position (starting at
 --   first Bit of primary header (Bit_Position: 1)) from Packet
 --   mapped to Integer/Unsigned_Integer.
 -- Parameters:
 --   OUT:
 --     Status:
 --       R_Success      retrieval was successful
 --       R_Range_Error  range violation
 --                      (Bit_Position+Bit_Length-1 > packet bit length)

8:   procedure Get_Raw_Real
      (in  Packet:           T_Packet;
       in  Bit_Position:     T_Position;
       out Value:            Real;
       out Status:           T_Raw_Status);

 -- Description:
 --   Retrieves 32 bits located at Bit_Position (starting at
 --   first Bit of primary header (Bit_Position: 1)) from Packet
 --   The output Value is mapped to Real.
 -- Parameters:
 --   OUT:
 --     Status:
 --       R_Success      retrieval was successful
 --       R_Range_Error  range violation
 --                      (Bit_Position+32-1 > packet bit length)
```

Copyright per DIN 34

```
 9:   procedure Get_Raw_Long_Real
       (in  Packet:           T_Packet;
        in  Bit_Position:     T_Position;
        out Value:            Long_Real;
        out Status:           T_Raw_Status);

  -- Description:
  --   Retrieves 64 bits located at Bit_Position (starting at
  --   first Bit of primary header (Bit_Position: 1)) from Packet
  --   The output Value mapped to Long_Real.
  -- Parameters:
  --   OUT:
  --     Status:
  --       R_Success      retrieval was successful
  --       R_Range_Error  range violation
  --                      (Bit_Position+64–1 > packet bit length)


 10:   procedure Get_Raw_Byte_Stream
       (in  Packet:           T_Packet;
        in  Bit_Position:     T_Position;
        in  No_Of_Bytes:      T_Byte_Length;
        out Value:            string of Byte;
        out Status:           T_Raw_Status);

  -- Description:
  --   Retrieves No_Of_Bytes word located at Bit_Position (starting
  --   first Bit of primary header (Bit_Position: 1)) from Packet
  --   The output Value is a string of bytes.
  -- Parameters:
  --   OUT:
  --     Status:
  --       R_Success      retrieval was successful
  --       R_Range_Error  range violation
  --                      (Bit_Position + (No_Of_Bytes)*8 – 1 > packet bit length)
  --       Invalid_Size:  Size of Value is invalid (< No_Of_Bytes)

 11:   procedure Get_Raw_Hex_Image
       (in  Packet:           T_Packet;
        in  Bit_Position:     T_Position;
        in  No_Of_Bytes:      T_Byte_Length;
        out Value:            string;
        out Status:           T_Raw_Status);

  -- Description:
  --   Retrieves No_Of_Bytes word located at Bit_Position (starting
  --   first Bit of primary header (Bit_Position: 1)) from Packet
  --   The output Value is a string containing the Hex–Image of
  --   the byte stream
  -- Parameters:
  --   OUT:
  --     Status:
  --       R_Success       retrieval was successful
  --       R_Range_Error   range violation
  --                       (Bit_Position + (No_Of_Bytes)*8 – 1 > packet bit length)
  --       R_Invalid_Size  Size of Value is invalid (< No_Of_Bytes)

 12:   procedure Convert_to_Hex_Image
       (in  Packet:           T_Packet;
        out Value:            string;
        out Status:           T_Raw_Status);

  -- Description:
  --   Converts Packet (string of byte) to a string containing
  --   the Hex–Image of the byte stream
  -- Parameters:
  --   OUT:
  --     Status:
  --       R_Success      retrieval was successful
  --       R_Invalid_Size Size of Value is invalid (< No_Of_Bytes)

-------------------------------------------------------------------------
end RAW_DATA_LIBRARY;
-------------------------------------------------------------------------
```

Copyright per DIN 34

I-11 UCL Ground System library: PACKET_library

The PACKET **library** contains operations to access a raw data packet

library Id (Body_Id): 13

I-11.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  PACKET_LIBRARY UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to maintain TM packet buffers
--     and Trigger APs
--
--     Must be compiled for ground and with Body Id = 13
--
--     ++++++++++++++++++++++++++++++++++++++++++++
--     CGS configuration parameter:
--     (see  $CGS_HOME/etc/configuration.xml)
--     ++++++++++++++++++++++++++++++++++++++++++++
--         TES.KERNEL.ADU.MAX_NUMBER_OF_ADU_BUFFER   :
--         -------------------------------
--             maximum number of open ADU Buffers. If this value is
--             reached, procedures Create_ADU_Packet_Buffer and
--             Start_AP_On_Arrival_Of_ADU will return  error status
--             P_Max_No_Of_Buffers_Reached
--
--         TES.KERNEL.ADU.BUFFER.MAX_NUMBER_OF_ADUS  :
--         -------------------------------
--             maximum number of packets stored in one Buffer.
--             If further ADUs arrive, they will be discarded by TES
--             until the next ADU was removed from Buffer (Get_Packet_From_Buffer)
--             The next packet will show the number of Packets discarded
--             in the out parameter <Packets_Discarded>
------------------------------------------------------------------------
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : PACKET_LIBRARY System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/packet_library_.ucl#6 $"
--*----------------------------------------------------------------
---
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------
-- Concept description and examples
------------------------------------------------------------------------
--
--  The PACKET_LIBRARY provides two libray procedures to establish an ADU buffer:
--
--     Create_ADU_Packet_Buffer and Start_AP_On_Arrival_Of_ADU
--
--  Both routines open an ADU-Buffer. The  parameter 'ADU' specifies the ADU which
--  will be stored in this buffer. The second procedure creates an ADU buffer with
--  a so-called "Trigger-AP" (defined by parameter AP_Pathname). This AP will be
--  started automatically if the specified ADU arrives. An AP-Pathname can be used
--  only once at a time, otherwise the command will fail with the error status
--  P_AP_Already_Used. Trigger APs must not have parameters. Otherwise an
--  'execution error' would be raised when the AP shall be started.  The parameter
--  Buffer_ID has to be used to switch between the buffer definitions and to access
--  the packets stored in the ADU buffer. Both routines may be called either from
--  UCL or from HLCL.
--
--  Further ADUs may be added to an existing buffer definition by the procedure
--  call:
--
--     Add_ADU_To_Packet_Buffer
--
--  Each ADU can be used in only one buffer. If the ADU is already used in a
--  packet buffer, a second "Create" or  "Add" command will fail with the Status
--  "P_ADU_Already_In_Buffer". The ADUs used in a packet buffer definitions
--  don't have to be acquired at definition time. If they are not "acquired" the
--  SAS will send no packets, but the buffer definition is valid. The SAS may be
--  stopped and restarted, this has no effect to the buffer definitions. Only
--  when TES is stopped and restarted, all buffers will be reset.
--
--  ADUs can also be removed from a buffer definition with the procedure call
--
--     Remove_ADU_From_Packet_Buffer
--
--  This procedure call will have no consequences to packets already stored in
--  the specified buffer.
--
```

Copyright per DIN 34

```
--  An ADU buffer with a Trigger-AP (defined by Start_AP_On_Arrival_Of_ADU) can
--  be changed into an ADU-Buffer without AP by calling the procedure
--
--     Remove_AP_From_ADU_Buffer
--
--  The maximum number of packet buffer managed by TES can be configured via a
--  CGS configuration parameter:
--
--  !! (U) The maximum number of ADU buffers. In this buffer TES put complete
--  !!     ADUs on request.
--  !!     Range: integer'range
--  !!     Recommended value: 5
--  TES.KERNEL.ADU.MAX_NUMBER_OF_ADU_BUFFER
--
--  The recommended number of buffers is 5. If this value is increased, the
--  memory size of the TES process will increase as well. If the maximum number
--  of buffers is already open, a newly Create_ADU_Packet_Buffer or
--  Start_AP_On_Arrival_Of_ADU command will fail with the error status
--  P_Max_No_Of_Buffers_Reached. UCL function calls provide information about
--  the number of buffers already defined, the number of free buffer definitions
--  still available as well as the number of packets stored in a certain buffer
--  and the number of free slots in a certain buffer. Further routines for the
--  listing of existing buffer definitions are described in section I-10.3.1.1.
--
--  The ADU Buffers in TES are static queues (FIFO: First In First Out). The
--  example buffer with the ID 15 contains 3 packets. The AP \CGS\READ_BUFFER
--  is going to retrieve a packet originally delivered by the SAS in an ADU_1.
--  After the call of "Get_Packet" (s. sect I-10.3.2) the ADU_1 packet will be
--  removed from the queue. The HLCL sequence is working with a different buffer
--  (Buffer 6), which contains only ADUs of type ADU_3 (currently 2 packets).
--  Buffer 23 will store 2 different ADUs (ADU_4 and ADU_5), but currently there
--  are no packets in the buffer. If one of those ADUs is delivered to TES the
--  AP \CGS\VERY_IMPORTANT will be started by TES.
--
--  The maximum number of packets to be stored in a buffer can be configured
--  via a CGS configuration parameter:
--
--  !! (U) The maximum ADUs per ADU buffer. (see above)
--  !!     Range: integer'range
--  !!     Recommended value: 20
--  TES.KERNEL.ADU.BUFFER.MAX_NUMBER_OF_ADUS
--
--  The recommended value is 20 ADUs per buffer.  If this value is increased,
--  the memory size of the TES process will increase as well. If a buffer
--  contains already the maximum number of packets in a buffer when a new packet
--  shall be stored, the packet will be discarded and the number of packets
--  discarded is incremented.  This is shown in figure I-10.3.1-Figure 2. When
--  the ADU_X was stored, the buffer was full. The next two packets were
--  discarded by TES. Afterwards some packets were removed (by call of "
--  Get_Packet") and when ADU_Y arrived, buffer space was available. When the
--  user is going to retrieve ADU_Y, the parameter Packets_Discarded will
--  indicate, that 2 packets were lost between ADU_X and ADU_Y.
--
--  A call of procedure
--
--     Close_Packet_Buffer
--
--  will close the buffer and discard all ADU packets still stored. If the same
--  Buffer_ID is also used in other APs, further actions with this AP will cause
--  P_Invalid_Buffer_ID, since Buffer_ID is no longer valid after the buffer has
--  been closed. To close all buffers at once the procedure
--
--     Reset_Packet_Buffers
--
--  may be used. This command should be used very carefully, since all buffers
--  will be reset.
--
---------------------------------------------
--  Buffer Definition Report Commands
---------------------------------------------
--
--  The Packet_Library provides also routines to get information about existing
--  buffer definitions. If the user wants to know, which Buffer-ID belongs to a
--  certain ADU, the function
--
--     Buffer_ID_For_ADU
--
--  gets this information. Analogue the function
--
--     Buffer_ID_For_AP
--
--  gets the Buffer-ID for a certain Trigger-AP. These Buffer-IDs could be used
--  to change or close the existing buffer definitions or to retrieve packets
--  The other way around is working the function
--
--     Trigger_AP
--
--  which delivers the Trigger-AP attached to a certain buffer definition.
--
--  If the ADUs belonging to a buffer definition shall be reported, the two
--  routines
```

Copyright per DIN 34

```
--
--     No_of_ADUs_for_Buffer, Buffer_ADU
--
--  may be used. No_Of_ADUs_for_Buffer returns the number of ADUs attached to a
--  buffer definition, and with Buffer_ADU the pathname of these ADUs can be get
--  sequentially.
--
--  If all buffer definitions shall be displayed, the two routines
--
--     No_Of_Buffers, Buffer_ID
--
--  are useful. Whereas No_of_Buffers returns the number of existing buffer
--  definitions, an iterator from 1..No_of_Buffers can be build with the
--  function Buffer_ID. The usage of these functions is shown in the example
--  HLCL sequence below:
--
--
--   ------------------------------------------------------------------------
--  | sequence BUFFER_DEF_REP;
--  |
--  |   variable no_buffers   : integer;
--  |   variable free_buffers : integer;
--  |   variable my_buffer_ID : integer;
--  |   variable adu_no       : integer;
--  |   variable number_of_packets : integer;
--  |   variable free_slots   : integer;
--  |
--  | begin
--  |
--  |   if ~IMPORTED(PACKET_LIBRARY) then
--  |     import PACKET_LIBRARY;
--  |   end if;
--  |
--  |   variable adu_path     : T_ADU_Path;
--  |   variable trigger_path : T_AP_Path;
--  |
--  |
--  |   put "";
--  |   put "================================================";
--  |   put "Packet Buffer Definition";
--  |   put "------------------------------------------------";
--  |
--  |   no_buffers := No_Of_Buffers;
--  |   free_buffers := No_Of_Free_Buffers;
--  |
--  |   put "Number of defined packet buffer: " + String(no_buffers) + " (free: " + String(free_buffers) + ")";
--  |   put "================================================";
--  |
--  |   for buffer_index := 1 to no_buffers do
--  |
--  |     put "";
--  |     my_buffer_ID := Buffer_ID(buffer_index);
--  |     put String(buffer_index) + ". Buffer-ID: " + String(my_buffer_ID);
--  |     adu_no := No_of_ADUs_for_Buffer(my_buffer_ID);
--  |     trigger_path := PACKET_LIBRARY.Trigger_AP(my_buffer_ID);
--  |     if trigger_path = \\ then
--  |       put "   No Trigger-AP";
--  |     else
--  |       put "   Trigger-AP: " + String(trigger_path);
--  |     end if;
--  |     put "   ADUs:";
--  |     for adu_index := 1 to adu_no do
--  |       adu_path := Buffer_ADU(my_buffer_ID, adu_index);
--  |       put "     " + String(adu_path);
--  |     end for; --adu_index := 1 to adu_no
--  |     number_of_packets := PACKETS_IN_BUFFER(buffer_index);
--  |     put "   Packets in Buffer    : " + String(number_of_packets);
--  |     free_slots        := No_Of_Free_Slots_In_Buffer(buffer_index);
--  |     put "   Free Slots in Buffer : " + String(free_slots);
--  |
--  |   end for; --  buffer_index := 1 to no_buffers
--  |
--  |   put "";
--  |   put "================================================";
--  |
--  |   delete adu_path;
--  |   delete trigger_path;
--  |
--  | end BUFFER_DEF_REP;
--   ------------------------------------------------------------------------
--
--  This sequence creates an output to the HLCL command window which looks like:
--
--  | HLCL: buffer_def_rep
--  |
--  | ================================================
--  | Packet Buffer Definition
--  | ------------------------------------------------
--  | Number of defined packet buffers: 4
--  | ================================================
--  |
```

Copyright per DIN 34

```
--  | 1. Buffer-ID: 94
--  |    No Trigger-AP
--  |    ADUs:
--  |      \CGS\COMMON\SAS\INT\TM\GMFE\ADU_TESTS\GMFE_1123_ALL
--  |      \CGS\COMMON\SAS\INT\TM\GMFE\ADU_TESTS\GMFE_1124_ALL
--  |
--  | 2. Buffer-ID: 95
--  |    Trigger-AP: \CGS\COMMON\SAS\RAW_DATA_ACCESS\TRIGGER_AP2
--  |    ADUs:
--  |      \CGS\COMMON\SAS\FE_SAS\GMFE\TM\ARCHIVE_ADU
--  |
--  | 3. Buffer-ID: 79
--  |    No Trigger-AP
--  |    ADUs:
--  |      \CGS\COMMON\SAS\INT\TM\GMFE\ADU_TESTS\GMFE_1123_ST1
--  |
--  | 4. Buffer-ID: 80
--  |    No Trigger-AP
--  |    ADUs:
--  |      \CGS\COMMON\SAS\INT\TM\GMFE\ADU_TESTS\GMFE_1123_ST2
--  |
--  | ================================================
--  | HLCL:
--
-------------------------------------------
-- Retrieval of Packets from ADU Buffer
-------------------------------------------
--
--  The retrieval of packets from buffer is done using the procedures
--
--     Get_Packet, Get_Full_Packet
--
--  Both procedures need the input parameter Buffer_ID and can therefore not be
--  used in a Trigger-AP (Trigger-AP has no visibility to the current Buffer_ID).
--  Packet retrieval from Trigger-APs is described below. The input parameter
--  Timeout specifies the duration until a packet has to arrive, in case no
--  packet is already in buffer. The default value Default_Timeout is 0.0
--  seconds, which means the packet has to be already in buffer, when the
--  procedure is called. If no packet is available until the timeout occurs,
--  the procedure call returns without packet and the Status P_Packet_Timeout.
--  Parameter Packet contains the packet delivered by ADU, but only if the
--  Status P_Success is returned also. The type of this parameter is "string of
--  byte". The evaluation of the packet may be done with the raw data library.
--
--  The procedure Get_Full_Packet  provides additional out parameters with
--  information about the ADU which contained the packet. ADU_Name returns the
--  MDB pathname of the ADU description. The type of ADU (stuctured,
--  unstructured, CCSDS_ADU, PUS_ADU) is available in parameter ADU_TYPE. The
--  timetag of the ADU packet (attached by the SAS)  is returned in parameter
--  Time_Tag. Whether the delivered Time is "Local Time" (LT) or "Simulated
--  Mission Time" (SMT) depends on a CGS configuration parameter:
--
--  !! (U) Indicates if the
--  !!     a) time stamp of the measurements, software variables
--  !!        and derived values            (accessible via UCL/HLCL get_...)
--  !!     b) time stamp of gdu acknowledge  (accessible via UCL/HLCL issue)
--  !!     c) time stamp of adu packet (accessible via UCL/HLCL get_full_packet...)
--  !!    is based on local time (true) or SMT (false).
--  !!    Range: boolean'range
--  !!    Recommended value: true
--  TES.KERNEL.DATA_PROCESSOR.TIME_STAMP_IN_LT
--
--  Note: This configuration parameter changes also the behaviour of other
--  system library procedures like 'Get_Integer', 'issue"  (GROUND_LIBRARY).
--
--  If a packet shall be retrieved from a Trigger-AP, the current Buffer_ID is
--  not available in the AP itself (since a Trigger_AP ist started by TES and
--  must not have parameters, the ID can not be passed via an AP parameter).
--  TES internally maps the AP-ID with the corresponding buffer. Therefore the
--  routines to be used from Trigger_APs only, do not have a parameter Buffer_ID:
--
--     Get_Packet_From_Buffer, Get_Full_Packet_From_Buffer
--
--  The functionality is similar to the two procedures described above. The
--  Trigger-AP procedures can not be used in HLCL. If a Trigger-AP is already
--  running when a new packet arrives, TES does not start the AP again. If the
--  AP has already finished, it will be started again. A "Single Shot Behaviour"
--  can be achieved, when the Trigger-AP closes the buffer before terminating.
--  Therefore another Close procedure without parameter Buffer_ID is provided:
--
--     Close_Buffer
--
--  If this command is used, the AP is started only once.
--
------------------------------------------------------------------------------

library PACKET_LIBRARY;

------------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------------
```

Copyright per DIN 34

```
  import RAW_DATA_LIBRARY;

-------------------------------------------------------------------------
-- TYPES
-------------------------------------------------------------------------

  type T_Buffer_ID = Unsigned_Integer;

  type T_Index = Integer(1..2147483647);

  type P_Status = (P_Success, P_Packet_Timeout, P_Invalid_Buffer_ID,
                   P_ADU_Unknown, P_ADU_Already_In_Buffer, P_Max_No_Of_Buffers_Reached,
                   P_ADU_Not_Registered, P_AP_Not_Registered, P_AP_Already_Used,
                   P_Other_Error);

  type T_AP_Path  = pathname (UCL_Automated_Procedure);

  type T_ADU_Path = pathname (ADU_Description);

  type T_ADU_Type = (CCSDS_ADU, PUS_ADU, ATV_ADU, Unstructured_ADU,
                     Structured_ADU, Undefined, Swarm_Adu);

  constant Default_Timeout:  Duration
    := 0.0 [s];
  constant Default_Buffer :   T_Buffer_ID
    := 0;

-------------------------------------------------------------------------
-- OPERATIONS
-------------------------------------------------------------------------

  1:  procedure Get_Status_Image
        (in  Status :   P_Status;
         out Image  :   string);

  -- Description:
  --   String-Conversion of P_Status

  2:  procedure Create_ADU_Packet_Buffer
        (in  ADU:                T_ADU_Path;
         out Buffer_ID:          T_Buffer_ID;
         out Status:             P_Status );

  -- Description:
  --   Definition of a packet buffer dependant on ADU
  --   (Buffer Size: See CGS configuration file)
  --   Further ADUs may be added with Add_ADU_To_Packet_Buffer
  -- Parameters:
  --   OUT:
  --     Status:
  --      P_Success:                  Buffer created successfully
  --      P_ADU_Unknown:              Enditem not maintained by testnode
  --      P_ADU_Already_In_Buffer:    ADU already used in a Buffer
  --      P_Max_No_Of_Buffers_Reached: max. number of Buffers reached
  --                                  (close buffers first)

  3:  procedure Add_ADU_To_Packet_Buffer
        (in  ADU:                T_ADU_Path;
         in  Buffer_ID:          T_Buffer_ID;
         out Status:             P_Status );

  -- Description:
  --   ADU will be added to Buffer definition
  -- Parameters:
  --   OUT:
  --     Status:
  --      P_Success:                  ADU added successfully
  --      P_Invalid_Buffer_ID:        Buffer not known
  --                                  (Create Buffer first)
  --      P_ADU_Unknown:              Enditem not maintained by testnode
  --      P_ADU_Already_In_Buffer:    ADU already used in a Buffer

  4:  procedure Remove_ADU_From_Packet_Buffer
        (in  ADU:                T_ADU_Path;
         in  Buffer_ID:          T_Buffer_ID;
         out Status:             P_Status );

  -- Description:
  --   ADU will be removed from Buffer definition
  --   This will habe no consequences to Packets already
  --   stored in Buffer
  -- Parameters:
  --   OUT:
  --     Status:
  --      P_Success:                  ADU removed successfully
  --      P_Invalid_Buffer_ID:        Buffer not known
  --                                  (Create Buffer first)
  --      P_ADU_Unknown:              Enditem not maintained by testnode
  --      P_ADU_Not_Registered:       ADU was not registered for Buffer
```

Copyright per DIN 34

```
5:  procedure Close_Packet_Buffer
      (in  Buffer_ID:        T_Buffer_ID;
       out Status:           P_Status);

-- Description:
--   Buffer will be closed and all packets dismissed.
--   Buffer definition no longer valid.
-- Parameters:
--   OUT:
--     Status:
--      P_Success:                  Buffer closed successfully
--      P_Invalid_Buffer_ID:        Buffer not known
--                                  (e.g. already closed)

6:  procedure Start_AP_On_Arrival_Of_ADU
      (in  ADU:                T_ADU_Path;
       in  AP_Pathname:        T_AP_Path();
       out Buffer_ID:          T_Buffer_ID;
       out Status:             P_Status );

-- Description:
--   Definition of a packet buffer dependant on ADU.
--   When a packet ADU has been arrived, the AP AP_Pathname
--   will be started (if not already running).
--   The so-called Trigger-AP has access to the packet buffer
--   (Get_Packet_From_Buffer, Get_Full_Packet_From_Buffer).
--   A Trigger-AP can be used only for one packet buffer.
--   If the AP is already attached to a buffer definition,
--   the procedure Start_AP_On_Arrival_Of_ADU returns the
--   Status P_AP_Already_Used.
--   Only parameterless APs are allowed for AP_Pathname. APs
--   with parameters cause an "execution error" at startup time.
--   Further ADUs may be added with Add_ADU_To_Packet_Buffer
--   The AP may be removed from the Buffer definition with
--   the procedure call Remove_AP_From_ADU_Buffer.
--   The Buffer created (Buffer_ID) may be closed by
--   Close_Packet_Buffer from ouside or by Close_Buffer from
--   inside a Trigger-AP.
--   If the Trigger-AP finishes without closing the buffer, the
--   buffer definition is still valid and the Trigger-AP will
--   be restarted when the next ADU packet arrives.
-- Parameters:
--   OUT:
--     Status:
--      P_Success:                  Buffer created successfully
--      P_ADU_Unknown:              Enditem not maintained by testnode
--      P_ADU_Already_In_Buffer:    ADU already used in a Buffer
--      P_Max_No_Of_Buffers_Reached: max. number of Buffers reached
--                                  (close buffers first)
--      P_AP_Already_Used:          AP_Pathname already in use for
--                                  Trigger

7:  procedure Close_Buffer(out Status: P_Status);

-- Description:
--   Trigger Buffer will be closed and all packets dismissed.
--   Buffer definition no longer valid.
--   This routine has to be used, if a packet buffer attached to
--   a Trigger AP shall be closed from the Trigger-AP itself.
--
-- Restriction:
--   To be used in UCL (Trigger-AP) only!
--
-- Parameters:
--   OUT:
--     Status:
--      P_Success:                  Buffer closed successfully
--      P_AP_Not_Registered         AP is not a Trigger-AP
--


8:  procedure Remove_AP_From_ADU_Buffer
      (in  Buffer_ID:          T_Buffer_ID;
       out Status:             P_Status );

-- Description:
--   This procedure call removes an AP from an ADU Buffer.
--   The Buffer definition is still valid, but no AP will
--   be started, if a new packet arrives.
-- Parameters:
--   OUT:
--     Status:
--      P_Success:                  AP removed from Buffer
--      P_Invalid_Buffer_ID:        Buffer not known
--                                  (e.g. already closed)
--      P_AP_Not_Registered:        No AP attached to Buffer



9:  procedure Get_Packet
      (in  Buffer_ID:          T_Buffer_ID;
```

Copyright per DIN 34

```
        in  Timeout:             Duration    := Default_Timeout;
        out Packet:              RAW_DATA_LIBRARY.T_Packet;
        out Packets_Discarded:   Integer;
        out Status:              P_Status);

10:  procedure Get_Full_Packet
       (in  Buffer_ID:           T_Buffer_ID;
        in  Timeout:             Duration    := Default_Timeout;
        out Packet:              RAW_DATA_LIBRARY.T_Packet;
        out Packets_Discarded:   Integer;
        out ADU_Name:            T_ADU_Path;
        out ADU_Type:            T_ADU_Type;
        out Time_Tag:            Time;
        out Status:              P_Status);

 -- Description (Get_Packet/Get_Full_Packet):
 --   Retrieves the next packet (packet will be removed from buffer) which
 --   is either already in buffer or arrives within Timeout. If
 --   Timeout is Default_Timeout (0.0 [s]), the packet has to be
 --   already in buffer.
 --   Note: If these procedures will be called from several APs
 --   or HLCL command windows with the same Buffer_ID, the result packets
 --   will be dispatched according to "first come – first serve"
 -- Parameters:
 --   OUT:
 --     Packets_Discarded:     packets since last procedure call
 --     Time_Tag:              the type of the returned Time_Tag, either local
 --                            time (LT) or Simulated Mission Time (SMT), depends
 --                            on the value of the TES variable
 --                            TES.KERNEL.DATA_PROCESSOR.TIME_STAMP_IN_LT (to be
 --                            defined via the CGS configuration file; see
 --                            configuration.xml).
 --     Status:
 --       P_Success:           Packet successfully retrieved
 --       P_Packet_Timeout:    No packet received within Timeout
 --       P_Invalid_Buffer_ID: Buffer ID is not valid (e.g. Buffer
 --                            already closed

11:  procedure Get_Packet_From_Buffer
       (in  Timeout:             Duration    := Default_Timeout;
        out Packet:              RAW_DATA_LIBRARY.T_Packet;
        out Packets_Discarded:   Integer;
        out Status:              P_Status);

12:  procedure Get_Full_Packet_From_Buffer
       (in  Timeout:             Duration    := Default_Timeout;
        out Packet:              RAW_DATA_LIBRARY.T_Packet;
        out Packets_Discarded:   Integer;
        out ADU_Name:            T_ADU_Path;
        out ADU_Type:            T_ADU_Type;
        out Time_Tag:            Time;
        out Status:              P_Status);

 -- Description (Get_Packet_From_Buffer/Get_Full_Packet_From_Buffer):
 --   The procedures Get_Packet_From_Buffer and Get_Full_Packet_From_Buffer
 --   must NOT be used outside from Trigger–APs. The belonging Buffer
 --   (defined by Start_AP_On_Arrival_Of_ADU) will be determined internally.
 --   The procedures retrieve the next packet (packet will be removed from buffer)
 --   which is either already in buffer or arrives within Timeout. If
 --   Timeout is Default_Timeout (0.0 [s]), the packet has to be
 --   already in buffer.
 --
 -- Restriction:
 --   To be used in UCL (Trigger–AP) only!
 --
 -- Parameters:
 --   OUT:
 --     Packets_Discarded:     packets since last procedure call
 --     Time_Tag:              the type of the returned Time_Tag, either local
 --                            time (LT) or Simulated Mission Time (SMT), depends
 --                            on the value of the TES variable
 --                            TES.KERNEL.DATA_PROCESSOR.TIME_STAMP_IN_LT (to be
 --                            defined via the CGS configuration file; see
 --                            configuration.xml).
 --     Status:
 --       P_Success:           Packet successfully retrieved
 --       P_Packet_Timeout:    No packet received within Timeout
 --       P_AP_Not_Registered: AP is not a Trigger–AP

13:  function Packets_In_Buffer
       (in  Buffer_ID:           T_Buffer_ID): Integer;

 -- Description:
 --   Returns number of packets still in buffer
 --   If Buffer_ID is invalid --> return value 0

14:  function No_Of_Free_Slots_In_Buffer
       (in  Buffer_ID:           T_Buffer_ID): Integer;

 -- Description:
 --   Returns number of free slots
```

Copyright per DIN 34

```
 --   If Buffer_ID is invalid --> return value 0

15:  function No_Of_Buffers : Integer;

 -- Description:
 --   Returns number of buffers defined

16:  function No_Of_Free_Buffers : Integer;

 -- Description:
 --   Returns number of buffers still available in TES
 --   (MAX_NUMBER_OF_ADU_BUFFER - No_Of_Buffers )

17:  procedure Reset_Packet_Buffers;

 -- Description:
 --   ALL Buffers will be closed and all packets dismissed.
 --   Buffer definitions no longer valid.
 -- CAUTION!!
 --   This command may cause P_Invalid_Buffer_ID errors
 --   in APs dealing with old packet buffer IDs!

18:  function Buffer_ID_For_ADU(ADU:T_ADU_Path) : T_Buffer_ID;

 -- Description:
 --   Returns Buffer_ID of buffer containing specified ADU
 --   If ADU not in a Buffer definition --> return value 0

19:  function Buffer_ID_For_AP(AP:T_AP_Path)   : T_Buffer_ID;

 -- Description:
 --   Returns Buffer_ID of buffer containing specified ADU
 --   If ADU not in a Buffer definition --> return value 0

20:  function Buffer_ID
      (Buffer_Index: T_Index) : T_Buffer_ID;

 -- Description:
 --   Returns Buffer_ID of Buffer_Index
 --   With this function, it is possible to iterate all
 --   buffer definitions
 --   Buffer_Index range is from 1 .. No_Of_Buffers
 --   If Buffer_Index > No_Of_Buffers --> return value 0

21:  function No_of_ADUs_for_Buffer
      (in  Buffer_ID:  T_Buffer_ID): Integer;

 -- Description:
 --   Returns number of ADU attached to specified buffer
 --   definition
 --   If Buffer_ID is invalid --> return value 0

22:  function Buffer_ADU
      (in  Buffer_ID:  T_Buffer_ID;
       in  Index:      T_Index): T_ADU_Path;

 -- Description:
 --   Returns ADU path attached to specified buffer definition
 --   With this function, it is possible to iterate the
 --   ADUs of a buffer definition
 --   Index range is from 1 .. No_of_ADUs_for_Buffer
 --   If Buffer_ID or Index  is invalid --> return value \\

23:  function Trigger_AP
      (in  Buffer_ID:  T_Buffer_ID): T_AP_Path;

 -- Description:
 --   Returns Trigger_AP path attached to specified buffer definition
 --   If Buffer_ID is invalid --> return value \\


-------------------------------------------------------------------------
end PACKET_LIBRARY;
-------------------------------------------------------------------------
```

Copyright per DIN 34

I-12 UCL Ground System library Specification: ATV_library

The **library** ATV_**library** contains operations to issue nested telecommands, to convert unsegmented time and to handle the sequence counter of TC's

library Id (Body_Id): 14

I-12.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ***************************************
--  ATV_LIBRARY UCL System Library
-- ***************************************
--
--ABSTRACT--
--     Defines Procedures to issue nested Telecommands, to
--     convert unsegmented time and to handle sequence count.
--
--     Must be compiled with Body Id = 14
--
--     recommended Nickname: "ATV_LIBRARY"
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : ATV_LIBRARY System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/atv_library_.ucl#6 $"
--*---------------------------------------------------------------------
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library ATV_LIBRARY;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

   import GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

  type SEQ_COUNT_RANGE = INTEGER (0 .. 2**14 - 1);

  type FINE_TIME_RANGE = INTEGER (0 .. 2**8  - 1);

  type CONSTRUCT_ITEM_NAME = PATHNAME (PUS_TC);

------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  1:  function CCSDS_UNSEGMENTED_TIME_TO_UCL_TIME
        (in  COARSE_TIME: UNSIGNED_INTEGER;
         in  FINE_TIME:   FINE_TIME_RANGE) : TIME;

  -- Description:
  --   This function returns the UCL time for the specified CCSDS unsegmented
  --   time (40 bits). The unsegmented time value is provided as 2 unsigned
  --   integer values. The base of the CCSDS unsegmented time is specified by
  --   CGS configuration parameter TES.KERNEL.GDU.TABLE.TAI_EPOCH_START_YEAR,
  --                               TES.KERNEL.GDU.TABLE.TAI_EPOCH_START_MONTH and
  --                               TES.KERNEL.GDU.TABLE.TAI_EPOCH_START_DAY,
  --   the default is 06.01.1980.
  -- Parameters:
  --   IN:
  --     COARSE_TIME         -- 32-Bit unsigned integer, seconds since TAI epoch
  --     FINE_TIME           -- 8-Bit unsigned integer


  2:  guarded procedure SET_SEQUENCE_COUNT
        (in  APPL:           APPLICATION_ID;
         in  SEQUENCE_COUNT: SEQ_COUNT_RANGE;
         out STATUS:         UCL_RETURN);

  -- Description:
  --   This operation sets a new start value of the sequence counter (per
  --   testnode) associated to the specified CCSDS packet application id.
  --   The given SEQUENCE_COUNT will be used for the next packet
```

Copyright per DIN 34

```
--   to send. After sending of a packet, the sequence count will
--   be incremented by one automatically.
--   The range of APPL is depend on CGS configuration parameter
--   TES.KERNEL.CCSDS.USE_PACKET_TYPE_TO_EXTEND_APID
--   (for more information – see $CGS_HOME/etc/configuration.xml).
-- Parameters:
--   IN:
--     APPL           -- the CCSDS packet application id
--     SEQUENCE_COUNT -- the value set to
--   OUT:
--     STATUS         -- UCL return status
--                         OK
--                         INVALID_PARAMETER

3:  function GET_SEQUENCE_COUNT
      (in APPL: APPLICATION_ID) : SEQ_COUNT_RANGE;

-- Description:
--   This function returns the actual sequence counter for the
--   specified CCSDS packet application id.
--   The returned sequence count will be used for the next packet
--   to send. The sequence count of the last packet is in general
--   GET_SEQUENCE_COUNT – 1
-- Parameters:
--   IN:
--     APPL           -- the CCSDS packet application id


4:  procedure ISSUE_TIME_TAGGED
      (in  EXTERNAL_TC:      CONSTRUCT_ITEM_NAME ();
       in  TIME_TAG:         UNSIGNED_INTEGER;
       in  INTERNAL_TC:      CONSTRUCT_ITEM_NAME ();
       out EXT_TC_SEQ_COUNT: SEQ_COUNT_RANGE;
       out INT_TC_SEQ_COUNT: SEQ_COUNT_RANGE;
       out SAS_ACK_CODE:     INTEGER;
       out SAS_ACK_TIME:     TIME;
       out STATUS:           UCL_RETURN);

-- Description:
--   The procedure ISSUE_TIME_TAGGED will construct a TC and send it to the
--   corresponding SAS of the external TC. This procedure construct the
--   external TC according to the specification stored in the database.
--   The time tag and the internal TC, which will constructed according to the
--   specification stored in the database too, will be inserted in the
--   datafield of the external TC.
-- Parameters:
--   IN:
--     EXTERNAL_TC        -- TC that will be emitted
--     TIME_TAG
-- onboard time
--     INTERNAL_TC        -- TC to be inserted in the external TC
--   OUT:
--     EXT_TC_SEQ_COUNT   -- sequence count of the external TC
--     INT_TC_SEQ_COUNT   -- sequence count of the internal TC
--     SAS_ACK_CODE       -- Status Code returned by SAS
--     SAS_ACK_TIME       -- Time when SAS acknowledged the command
--     STATUS             -- UCL return status
--                           OK
--                           INVALID_TESTNODE_MODE;
--                           INVALID_ITEM_NAME;
--                           ITEM_IS_DISABLED;
--                           NO_GDU_SERVICE;
--                           INVALID_APPLICATION_NAME;
--                           COMMUNICATION_ERROR;
--                           APPLICATION_NACK;
--                           TIMEOUT;
--                           AUTHORIZATION_FAILED;
--                           INVALID_PARAMETER;
--                           NOT_OK;
--                           RUNTIME_ERROR;
--                           PRECONDITION_CHECK_FAILED;
--                           APPLICATION_NOT_READY;


END ATV_LIBRARY;
```

Copyright per DIN 34

I-13 UCL Ground System library Specification: TEXT_FILE_IO

The **library** TEXT_FILE_IO contains operations for input and output to text files.

library Id (Body_Id): 15

I-13.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  TEXT_FILE_IO UCL System Library
-- ****************************************
--
-- ABSTRACT
--     Defines Functions and procedures for access
--     and basic handling of text files.
--
--     Must be compiled for ground and with Body Id = 15
--
--     recommended Nickname: "TEXT_FILE_IO"
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : TEXT_FILE_IO System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/text_file_io_.ucl#7 $"
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library TEXT_FILE_IO;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

   import GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type BASE_RANGE = UNSIGNED_INTEGER (2 .. 16);

   type TEXT_FILE_HANDLE = WORD;

------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------

   constant HANDLE_IN_USE : UCL_RETURN := 200;

   -- temporary solution for SPR-16944

   -- This return values is given for procedure open, create or append,
   -- if this file was allready opened and the handle may be in use.


------------------------------------------------------------------------
-- PARAMETER DESCRIPTION:
------------------------------------------------------------------------

   --
   -- Description:
   --
   --   Note:  This parameter description is valid for all operations below.
   --
   -- Parameters:
   --   IN:
   --     NAME     : the file name
   --     FILE     : handle to text file
   --     ITEM     : item to write
   --     WIDTH    : output width of current item in character
   --                0 means minimum needed width
   --     BASE     : base
   --     AFT      : character count for character after '.'
   --                0 means default aft depend on type
   --     EXP      : character count for exponent
   --                0 means default EXP depend on type
   --     GROUPING : a space after GROUPING characters
   --                0 means no grouping
```

Copyright per DIN 34

```
   --     COUNT    : count of read characters
   --     COLUMN   : column
   --   OUT:
   --     FILE     : handle to text file
   --     NAME     : the file name
   --     STATUS   : UCL return status from GROUND_COMMON
   --        OK:                  UCL_RETURN :=   1; for success of operation
   --        NOT_OK:              UCL_RETURN :=  57; for any other failure
   --        NO_FREE_HANDLE :     UCL_RETURN := 124; when all handles used up
   --
   --        EXISTING_FILE :      UCL_RETURN := 127; when trying to open an open file
   --                                                and the modes are not compatible
   --        FILE_NOT_FOUND:      UCL_RETURN := 128; when trying to open non-existing
   --                                              file or new file in non-existing
   --                                              directory
   --        FILE_NOT_OPEN:       UCL_RETURN := 129; when accessing not yet opened file
   --        FILE_DATA_ERROR:     UCL_RETURN := 130; when reading from text file fails
   --
   --        NOT_YET_IMPLEMENTED : UCL_RETURN := 103; when not yet implemented

-------------------------------------------------------------------------
-- OPERATIONS
-------------------------------------------------------------------------


   -------------
   -- general --
   -------------

   --   (1) Only files may be accessed on this partitions,
   --       which are accessible by the TES process.
   --   (2) Only files with the correct permissions are accessible
   --       (i.e. cgsadmin).
   --   (3) The created files inherit the user_id, group_id and the
   --       permissions of the user of the running TES process
   --       (i.e. cgsadmin).
   --

   ----------------------------
   -- File access operations --
   ----------------------------

   -- rules for concurrent file access
   --
   --  Allow the concurrent access for writing and reading a file.
   --  Do not allow the creation of a file, which was opened by calls APPEND or OPEN.
   --
   --  In case of call
   --
   --    A) CREATE for a file, which was opened by call CREATE
   --    B) OPEN   for a file, which was opened by call OPEN
   --    C) APPEND for a file, which was opened by call APPEND
   --
   --  CGS returns the existing file handle and set status to Handle_In_Use (200).
   --  In these cases, the system do not set the file pointer correct, or reset file
   --  content!
   --
   --  In case of call
   --
   --    D) CREATE for a file, which was opened by call APPEND or OPEN
   --    E) APPEND for a file, which was opened by call CREATE
   --
   --  CGS returns the status Existing_File (127).
   --
   --  In case of call
   --
   --    F) OPEN   for a file, which was opened by call CREATE or APPEND
   --    G) APPEND for a file, which was opened by call OPEN
   --
   --  CGS returns a new file handle and status Ok (1).
   --

  1:   procedure CREATE          (in  NAME     : STRING           := "";
                                  out FILE     : TEXT_FILE_HANDLE;
                                  out STATUS   : UCL_RETURN  );
   -- Description:
   --   Create new file for output, replacing existing file.
   -- Parameters:
   --   IN:
   --     NAME : the file name to be craete for output.
   --            "" means temporary file.

  2:   procedure OPEN            (in  NAME     : STRING;
                                  out FILE     : TEXT_FILE_HANDLE;
                                  out STATUS   : UCL_RETURN);
   -- Description:
   --   Open existing file for input.


  3:   procedure APPEND          (in  NAME     : STRING;
                                  out FILE     : TEXT_FILE_HANDLE;
```

Copyright per DIN 34

```
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   Open existing file to append output.

 4:   procedure CLOSE           (in  out FILE : TEXT_FILE_HANDLE;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   Close opened file.

 5:   procedure RESET           (in  FILE     : TEXT_FILE_HANDLE;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   Set current position to beginning of file for input.

 6:   procedure REWRITE         (in  FILE     : TEXT_FILE_HANDLE;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   Set current position to beginning of file for output, overwriting.

 7:   procedure REMOVE          (in  FILE     : TEXT_FILE_HANDLE;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   Delete previously opened file.

  ----------------------------
  --   File status operations --
  ----------------------------

 8:   function  END_OF_FILE (in  FILE     : TEXT_FILE_HANDLE): BOOLEAN;
  -- Description:
  --   Return TRUE if current position is at EOF.


 9:   function  END_OF_LINE (in  FILE     : TEXT_FILE_HANDLE): BOOLEAN;
  -- Description:
  --   Return TRUE if current position is at EOL.

10:   function  IS_OPEN (in  FILE     : TEXT_FILE_HANDLE): BOOLEAN;
  -- Description:
  --   Return TRUE if File represents an opened file.

11:   function  FILE_SIZE
    (in  FILE     : TEXT_FILE_HANDLE ): UNSIGNED_INTEGER;
  -- Description:
  --   Return size of file in bytes.


12:   procedure GET_FILE_NAME   (in  FILE     : TEXT_FILE_HANDLE;
                                out NAME     : STRING);
  -- Description:
  --   Return name of file represented by File.

  ----------------------------
  -- Text Output operations  --
  ----------------------------


13:   procedure WRITE          (in  FILE     : TEXT_FILE_HANDLE;
                                in  ITEM     : STRING;
                                in  WIDTH    : INTEGER          := 0;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   write no more than width characters to text file without line feed


14:   procedure WRITE_LINE     (in  FILE     : TEXT_FILE_HANDLE;
                                in  ITEM     : STRING           := "";
                                in  WIDTH    : INTEGER          := 0;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   write no more than width characters to text file with line feed


15:   procedure WRITE_CHARACTER (in  FILE     : TEXT_FILE_HANDLE;
                                in  ITEM     : CHARACTER;
                                in  WIDTH    : INTEGER          := 0;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   write character to text file without line feed


16:   procedure WRITE_INTEGER   (in  FILE     : TEXT_FILE_HANDLE;
                                in  ITEM     : INTEGER;
                                in  WIDTH    : INTEGER          := 0;
                                in  BASE     : BASE_RANGE       := 10;
                                out STATUS   : UCL_RETURN);
  -- Description:
  --   write integer to text file without line feed


17:   procedure WRITE_UNSIGNED  (in  FILE     : TEXT_FILE_HANDLE;
```

Copyright per DIN 34

```
                              in  ITEM     : UNSIGNED_INTEGER;
                              in  WIDTH    : INTEGER          := 0;
                              in  BASE     : BASE_RANGE       := 10;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write unsigned integer to text file without line feed


18:   procedure WRITE_REAL      (in  FILE     : TEXT_FILE_HANDLE;
                              in  ITEM     : REAL;
                              in  WIDTH    : INTEGER          := 0;
                              in  AFT      : UNSIGNED_INTEGER := 0;
                              in  EXP      : UNSIGNED_INTEGER := 0;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write real to text file without line feed


19:   procedure WRITE_LONG_REAL (in  FILE     : TEXT_FILE_HANDLE;
                              in  ITEM     : LONG_REAL;
                              in  WIDTH    : INTEGER          := 0;
                              in  AFT      : UNSIGNED_INTEGER := 0;
                              in  EXP      : UNSIGNED_INTEGER := 0;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write long real to text file without line feed


20:   procedure WRITE_STATECODE (in  FILE     : TEXT_FILE_HANDLE;
                              in  ITEM     : STATECODE;
                              in  WIDTH    : INTEGER          := 0;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write statecode to text file without line feed


21:   procedure WRITE_TIME      (in  FILE     : TEXT_FILE_HANDLE;
                              in  ITEM     : TIME;
                              in  WIDTH    : INTEGER          := 0;
                              in  AFT      : UNSIGNED_INTEGER := 0;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write time to text file without line feed


22:   procedure WRITE_BYTES     (in  FILE     : TEXT_FILE_HANDLE;
                              in  ITEM     : STRING of BYTE;
                              in  WIDTH    : INTEGER          := 0;
                              in  GROUPING : UNSIGNED_INTEGER := 0;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   write bytes to text file without line feed

  ----------------------------
  -- Text Input operations --
  ----------------------------

23:   procedure Read            (in  FILE     : TEXT_FILE_HANDLE;
                              in  COUNT    : UNSIGNED_INTEGER;
                              out ITEM     : STRING;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   read no more than Count characters from text file (up to EOL)

24:   procedure Read_Line       (in  FILE     : TEXT_FILE_HANDLE;
                              out ITEM     : STRING;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   read Text line from text file


25:   procedure Read_Integer    (in  FILE     : TEXT_FILE_HANDLE;
                              out Item     : Integer;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   read Integer value from text file


26:   procedure Read_Unsigned   (in  FILE     : TEXT_FILE_HANDLE;
                              out ITEM     : UNSIGNED_INTEGER;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   read Unsigned value from text file


27:   procedure Read_Real       (in  FILE     : TEXT_FILE_HANDLE;
                              out Item     : Real;
                              out STATUS   : UCL_RETURN);
  -- Description:
  --   read real value from text file
```

Copyright per DIN 34

```
28:   procedure Read_Long_Real  (in  FILE     : TEXT_FILE_HANDLE;
                                 out ITEM     : LONG_REAL;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read long_real value from text file


29:   procedure Read_Statecode  (in  FILE     : TEXT_FILE_HANDLE;
                                 out ITEM     : STATECODE;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read statecode value from text file


30:   procedure Read_Time       (in  FILE     : TEXT_FILE_HANDLE;
                                 out ITEM     : TIME;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read time value from text file


31:   procedure Read_Bytes      (in  FILE     : TEXT_FILE_HANDLE;
                                 in  COUNT    : UNSIGNED_INTEGER;
                                 out ITEM     : STRING of BYTE;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read no more than Count bytes from text file (or up to EOF)

  ------------------------------------------------------
  -- Text Positioning operations (input and output) --
  ------------------------------------------------------

32:   function  File_Column
      (in  FILE     : TEXT_FILE_HANDLE ): UNSIGNED_INTEGER;
  -- Description:
  --   Get current column position for text file

33:   procedure Set_File_Column (in  FILE     : TEXT_FILE_HANDLE;
                                 in  COLUMN   : UNSIGNED_INTEGER;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   Specifiy next column position for text file


end TEXT_FILE_IO;
```

Copyright per DIN 34

I-14 UCL Ground System library Specification: BINARY_FILE_IO

The **library** BINARY_FILE_IO contains operations for input and output to bimary files.

library Id (Body_Id): 16

I-14.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  BINARY_FILE_IO UCL System Library
-- ****************************************
--
-- ABSTRACT
--     Defines Functions and procedures for access
--     and basic handling of binary data files.
--
--     Must be compiled for ground and with Body Id = 16
--
--     recommended Nickname: "BINARY_FILE_IO"
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : BINARY_FILE_IO System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/binary_file_io_.ucl#6 $"
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library BINARY_FILE_IO;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

   import GROUND_COMMON;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type BINARY_FILE_HANDLE = WORD;

------------------------------------------------------------------------
-- CONSTANTS
------------------------------------------------------------------------

   constant HANDLE_IN_USE : UCL_RETURN := 200;

   -- temporary solution for SPR-16944

   -- This return values is given for procedure open, create or append,
   -- if this file was allready opened and the handle may be in use.


------------------------------------------------------------------------
-- PARAMETER DESCRIPTION:
------------------------------------------------------------------------

   --
   -- Description:
   --
   --   Note:  This parameter description is valid for all operations below.
   --
   -- Parameters:
   --   IN:
   --     NAME     : the file name
   --     FILE     : handle to binary file
   --     ITEM     : item to write
   --     COUNT    : count of read characters
   --   OUT:
   --     FILE     : handle to binary file
   --     NAME     : the file name
   --     STATUS   : UCL return status from GROUND_COMMON
   --        OK:                  UCL_RETURN :=   1; for success of operation
   --        NOT_OK:              UCL_RETURN :=  57; for any other failure
   --        NO_FREE_HANDLE :     UCL_RETURN := 124; when all handles used up
   --
   --        EXISTING_FILE :      UCL_RETURN := 127; when trying to open an open file
   --                                                and the modes are not compatible
```

Copyright per DIN 34

```
   --        FILE_NOT_FOUND:       UCL_RETURN := 128; when trying to open non-existing
   --                                              file or new file in non-existing
   --                                              directory
   --        FILE_NOT_OPEN:        UCL_RETURN := 129; when accessing not yet opened file
   --        FILE_DATA_ERROR:      UCL_RETURN := 130; when reading from text file fails
   --
   --        NOT_YET_IMPLEMENTED : UCL_RETURN := 103; when not yet implemented

-------------------------------------------------------------------------
-- OPERATIONS
-------------------------------------------------------------------------

   -------------
   -- general --
   -------------

   --   (1) Only files may be accessed on this partitions,
   --       which are accessible by the TES process.
   --   (2) Only files with the correct permissions are accessible
   --       (i.e. cgsadmin).
   --   (3) The created files inherit the user_id, group_id and the
   --       permissions of the user of the running TES process
   --       (i.e. cgsadmin).
   --

   ----------------------------
   -- File access operations --
   ----------------------------

   -- rules for concurrent file access
   --
   --  Allow the concurrent access for writing and reading a file.
   --  Do not allow the creation of a file, which was opened by calls APPEND or OPEN.
   --
   --  In case of call
   --
   --    A) CREATE for a file, which was opened by call CREATE
   --    B) OPEN   for a file, which was opened by call OPEN
   --    C) APPEND for a file, which was opened by call APPEND
   --
   --  CGS returns the existing file handle and set status to Handle_In_Use (200).
   --  In these cases, the system do not set the file pointer correct, or reset file
   --  content!
   --
   --  In case of call
   --
   --    D) CREATE for a file, which was opened by call APPEND or OPEN
   --    E) APPEND for a file, which was opened by call CREATE
   --
   --  CGS returns the status Existing_File (127).
   --
   --  In case of call
   --
   --    F) OPEN   for a file, which was opened by call CREATE or APPEND
   --    G) APPEND for a file, which was opened by call OPEN
   --
   --  CGS returns a new file handle and status Ok (1).
   --

  1:   procedure CREATE      (in  NAME     : STRING           := "";
                              out FILE     : BINARY_FILE_HANDLE;
                              out STATUS   : UCL_RETURN);
    -- Description:
    --   Create new file for output, replacing existing file:
    -- Parameters:
    --   IN:
    --     NAME : the file name to be craete for output.
    --            "" means temporary file.

  2:   procedure OPEN        (in  NAME     : STRING;
                              out FILE     : BINARY_FILE_HANDLE;
                              out STATUS   : UCL_RETURN);
    -- Open existing file for input:
    -- Description:

  3:   procedure APPEND      (in  NAME     : STRING;
                              out FILE     : BINARY_FILE_HANDLE;
                              out STATUS   : UCL_RETURN);
    -- Description:
    --   Open existing file to append output:

  4:   procedure CLOSE       (in  out FILE : BINARY_FILE_HANDLE;
                              out STATUS   : UCL_RETURN);
    -- Description:
    --   Close opened file:

  5:   procedure RESET       (in  FILE     : BINARY_FILE_HANDLE;
                              out STATUS   : UCL_RETURN);
    -- Description:
    --   Set current position to beginning of file for input:
```

Copyright per DIN 34

```
 6:   procedure REWRITE         (in  FILE     : BINARY_FILE_HANDLE;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   Set current position to beginning of file for output, overwriting:

 7:   procedure REMOVE      (in  FILE     : BINARY_FILE_HANDLE;
                             out STATUS   : UCL_RETURN);
  -- Description:
  --   Delete previously opened file:

  ----------------------------
  -- File status operations --
  ----------------------------

 8:   function  END_OF_FILE (in  FILE : BINARY_FILE_HANDLE ): BOOLEAN;
  -- Description:
  --   Return TRUE if current position is at EOF:

 9:   function  IS_OPEN (in  FILE : BINARY_FILE_HANDLE ): BOOLEAN;
  -- Description:
  --   Return TRUE if File represents an opened file:

10:   function  FILE_SIZE (in  FILE : BINARY_FILE_HANDLE ): UNSIGNED_INTEGER;
  -- Description:
  --   Return size of file in bytes:

11:   procedure GET_FILE_NAME (in  FILE     : BINARY_FILE_HANDLE;
                              out NAME     : STRING);
  -- Description:
  --   Return name of file represented by File:

  -------------------------------
  -- Binary Output operations --
  -------------------------------
  -- Binary files are written and read in the MSB
  -- Network format (architecture-independent)

12:   procedure WRITE_BYTE  (in  FILE     : BINARY_FILE_HANDLE;
                             in  ITEM     : BYTE;
                             out STATUS   : UCL_RETURN);
  -- Description:
  --   write a single byte to binary file

13:   procedure WRITE_WORD  (in  FILE     : BINARY_FILE_HANDLE;
                             in  ITEM     : WORD;
                             out STATUS   : UCL_RETURN);
  -- Description:
  --   write a single Word to binary file

14:   procedure WRITE_LONG_WORD (in  FILE     : BINARY_FILE_HANDLE;
                                 in  ITEM     : LONG_WORD;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   write a single Long_Word to binary file

15:   procedure WRITE_RAW_BYTES (in  FILE     : BINARY_FILE_HANDLE;
                                 in  ITEM     : STRING of BYTE;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   write a byte string to binary file

  ----------------------------
  -- Binary Input operations --
  ----------------------------

16:   procedure READ_BYTE   (in  FILE     : BINARY_FILE_HANDLE;
                             out ITEM     : BYTE;
                             out STATUS   : UCL_RETURN);
  -- Description:
  --   read a single byte from binary file

17:   procedure READ_WORD    (in  FILE     : BINARY_FILE_HANDLE;
                             out ITEM     : WORD;
                             out STATUS   : UCL_RETURN);
  -- Description:
  --   read a single Word from binary file

18:   procedure READ_LONG_WORD  (in  FILE     : BINARY_FILE_HANDLE;
                                 out ITEM     : LONG_WORD;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read a single Long_Word from binary file

19:   procedure READ_RAW_BYTES  (in  FILE     : BINARY_FILE_HANDLE;
                                 in  COUNT    : UNSIGNED_INTEGER;
                                 out ITEM     : STRING of BYTE;
                                 out STATUS   : UCL_RETURN);
  -- Description:
  --   read no more than Count bytes from binary file (or up to EOF)

end BINARY_FILE_IO;
```

Copyright per DIN 34

I-15 UCL Ground System library Specification: SYSTEM

The **library** SYSTEM contains some Operation System specific operations.

library Id (Body_Id): 17

I-15.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- **************************
--  System UCL System Library
-- **************************
--
-- ABSTRACT
--     Defines operations for OS command execution.
--
--     Must be compiled for ground and with Body Id = 17
--
--     recommended Nickname: "SYSTEM"
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : System System-Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/system_.ucl#5 $"
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library SYSTEM;

------------------------------------------------------------------------
-- IMPORTS
------------------------------------------------------------------------

   import GROUND_COMMON;

------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  ----------------------
  -- Command execution --
  ----------------------

  1:  guarded procedure EXECUTE
        (in  COMMAND     : STRING;
         in  PARAMETER   : STRING := "";
         in  NODE        : NODE_NAME := \\;
         out PID         : UNSIGNED_INTEGER;
         out STATUS      : UCL_RETURN);

  -- Description:
  --   This operation creates a new process and executes COMMAND
  --   using '/bin/sh' together with the specified parameter on the specified
  --   node. The created process will run completely independent.
  --   It is up to the user to redirect input and output if necessary.
  --   The started process is running with the user_id und group_id
  --   of the user of the running PCS process (i.e. cgsadmin).
  -- Parameters:
  --   IN:
  --     COMMAND    : command to be exceute
  --     PARAMETER  : parameter list
  --     NODE       : Node on which the command is to be executed, by default
  --                  the current node.
  --   OUT:
  --     PID        : OS process identifier of the created process.
  --     STATUS     : UCL return status
  --                    OK
  --                    NOT_OK

  2:  guarded procedure EXECUTE_AND_WAIT
        (in  COMMAND     : STRING;
         in  PARAMETER   : STRING := "";
         in  NODE        : NODE_NAME := \\;
         in  TIMEOUT     : DURATION;
         out PID         : UNSIGNED_INTEGER;
         out UNIX_RETURN : INTEGER;
         out STATUS      : UCL_RETURN);

  -- Description:
```

Copyright per DIN 34

```
  --   This operation creates a new process and executes COMMAND
  --   using '/bin/sh' together with the specified parameter on the specified
  --   node. The created process will run synchronized, i.e. control will
  --   be returned to the caller upon termination of the created process or
  --   upon timeout expiration.
  --   The started process is running with the user_id und group_id
  --   of the user of the running PCS process (i.e. cgsadmin).
  -- Parameters:
  --   IN:
  --     COMMAND    : command to be exceute
  --     PARAMETER  : parameter list
  --     NODE       : Node on which the command is to be executed, by default
  --                  the current node.
  --   OUT:
  --     PID        : OS process identifier of the created process.
  --                  This parameter is valid only, if STATUS = TIMEOUT,
  --                  ie. the process is still running.
  --     UNIX_RETURN: execution result of the created process
  --                  This parameter is valid only, if STATUS = OK
  --     STATUS     : UCL return status
  --                   OK
  --                   TIMEOUT
  --                   NOT_OK

  ---------------------
  -- File operations --
  ---------------------

  -------------
  -- general --
  -------------

  --   (1) Only files, directories may be accessed on this partitions,
  --       which are accessible by the TES process.
  --   (2) Only files, directories with the correct permissions are
  --       accessible (i.e. cgsadmin).
  --   (3) The created files, directories inherit the user_id,
  --       group_id and the permissions of the user of the running
  --       TES process (i.e. cgsadmin).

  3:  function  EXISTS
        (in  NAME : STRING): BOOLEAN;

  -- Description:
  --   This operation returns TRUE if specified file exists.
  -- Parameters:
  --   IN:
  --     NAME   : file to be checked

  4:  guarded procedure DELETE
        (in  NAME   : STRING;
         out STATUS : UCL_RETURN);

  -- Description:
  --   This operation deletes the specified file.
  -- Parameters:
  --   IN:
  --     NAME   : file to be delete
  --   OUT:
  --     STATUS : UCL return status
  --                   OK
  --                   NOT_OK
  --                   FILE_NOT_FOUND

  5:  procedure CREATE_DIRECTORY
        (in  NAME   : STRING;
         out STATUS : UCL_RETURN);

  -- Description:
  --   This operation creates the specified directory.
  --   Is identical to unix call 'mkdir -p'.
  -- Parameters:
  --   IN:
  --     NAME   : directory to be create
  --   OUT:
  --     STATUS : UCL return status
  --                   OK
  --                   NOT_OK

end SYSTEM;
```

Copyright per DIN 34

I-16 UCL Ground System library Specification: CALIBRATION_LIB

The **library** CALIBRATION_LIB contains **procedures** used for access to calibration description of measurement values. This **library** replace the calibration part of the ground_values **library**.

library Id (Body_Id): 18

I-16.1 UCL System library Specification

```
------------------------------------------------------------------------
--
-- ****************************************
--  CALIBRATION UCL System Library
-- ****************************************
--
--ABSTRACT--
--     Defines Procedures to calibrate/decalibrate acquired values in CGS
--     Must be compiled for ground and with Body Id = 18
--
--IDENTIFICATION--
--      PROJECT NAME : CGS
--      OBJECT NAME  : CALIBRATION System Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/calibration_lib_.ucl#7 $"
------------------------------------------------------------------------
---
--CONTENTS--
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
--CHANGE HISTORY
--
--END HISTORY
--
------------------------------------------------------------------------

library CALIBRATION_LIB;

  ----------------------------------------
  --  Calibration / Decalibration Functions
  ----------------------------------------
  --  Calibration means conversion from raw value to engineering value.
  --  Decalibration means conversion from engineering value to raw value.
  --  Measurements have a calibration function, while stimuli
  --  (i.e. their parameters) have a decalibration function assigned
  --  Measurements: CALIBRATION: raw value -> engineering value
  --  Stimulus/TC Parameter: DECALIBRATION: enginering value -> raw value
  --
  --   Note: Measurements bear a calibration function, while stimuli (i.e.
  --     their parameter)  bear a decalibration function. Thus for  measurements,
  --     the function specified in the MDB is converted to a decalibration
  --     function and then applied. For stimuli, the function specified in the
  --     MDB is used to calculate RAW_VALUE from the ENG_VALUE
  --
  --   Measurement: CALIBRATION   info -> convert -> apply
  --   Stimulus:    DECALIBRATION info -> apply
  --
  --   Note: CGS allows following measurement calibration combinations:
  --
  --                |                 eng value
  --                |-------------------------------------------------
  --      raw value  | state| integer | unsigned | real  | long  | byte
  --                 | code |         |          |       | real  | stream
  --     ------------|------|---------|----------|-------|-------|--------
  --                 |      |         |          |       |       |
  --     integer     |      |    X    |    X     |   X   |   X   |   X
  --                 |      |         |          |       |       |
  --     unsigned    |   X  |    X    |    X     |   X   |   X   |   X
  --                 |      |         |          |       |       |
  --     real        |      |    X    |    X     |   X   |   X   |
  --                 |      |         |          |       |       |
  --     long real   |      |    X    |    X     |   X   |   X   |
  --                 |      |         |          |       |       |
  --     byte stream |      |         |          |       |       |   X
  --                 |      |         |          |       |       |
  --

  import GROUND_COMMON; -- reference by nickname

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

  type POINT_PAIRS = record
```

Copyright per DIN 34

```
    RAW_VALUE         : LONG_REAL;
    ENGINEERING_VALUE : LONG_REAL;
  end record;

  type STATECODE_PAIRS = record
    RAW_VALUE         : UNSIGNED_INTEGER;
    HIGH_RAW_VALUE    : UNSIGNED_INTEGER;
    ENGINEERING_VALUE : STATECODE;
  end record;

  type INTEGER_STRING_PAIRS = record
    RAW_VALUE         : INTEGER;
    HIGH_RAW_VALUE    : INTEGER;
    ENGINEERING_VALUE : VALUE_STRING;
  end record;

  type UNSIGNED_STRING_PAIRS = record
    RAW_VALUE         : UNSIGNED_INTEGER;
    HIGH_RAW_VALUE    : UNSIGNED_INTEGER;
    ENGINEERING_VALUE : VALUE_STRING;
  end record;

  type CALIBRATION_TYPE = (UNDEFINED, IDENTICAL, POLYNOM,
                           POINT_PAIR, DISCRETE, SUBSTRING,
                           INTEGER_STRING, UNSIGNED_STRING);

-------------------------------------------------------------------------
-- PROCEDURES & FUNCTIONS
-------------------------------------------------------------------------


  --------------------------
  --  Calibration Functions
  --------------------------

 ------------------------------------------
 --      Set/Get Calibration for Measurments
 ------------------------------------------

  1:  function GET_CALIBRATION_TYPE
        (in  ITEM: ENDITEM_WITH_RAW_VALUE) : CALIBRATION_TYPE;

  -- Description
  --   Gets the calibration type of an enditem.
  --   CALIBRATION_TYPE NONE is returned in error case.
  --
  --   Note: For all (and only for this) Calibration Types = POLYNOM with
  --         calibration coefficent = (0.0, 1.0, 0.0, 0.0, 0.0)
  --         this funtion return "IDENTICAL".
  --
  -- Parameters:
  --   IN:
  --     ITEM:    enditem to be changed the calibration description

  2:  guarded procedure SET_CALIBRATION_COEFFICIENT -- for polynomial calibration
        (in  ITEM               : ENDITEM_WITH_RAW_VALUE;
         in  COEFFICIENT_NUMBER : COEFFICIENT_RANGE;
         in  COEFFICIENT_VALUE  : LONG_REAL;
         out STATUS             : UCL_RETURN);

  -- Description
  --     Sets one polynomial coefficient for the calibration definition of an enditem
  --     Type of calibration is automatically changed to "POLYNOM"
  -- Parameters:
  --   IN:
  --     ITEM                : Enditem to be changed the calibration description
  --     COEFFICIENT_NUMBER   : Number of the coefficient
  --     COEFFICIENT_VALUE    : New value of the coefficient
  --   OUT:
  --     STATUS returns the UCL Return Code values as defined in GROUND_COMMON:
  --      OK                  : New Coefficient applied
  --      ACQUISITION_ENABLED : Calibration cannot be changed if acquisition of item is
  --                            already on
  --      CAL_ERROR           : An exception occured when setting the new coefficient
  --      ITEM_UNKNOWN        : Enditem is not known on the executing node


  3:  procedure GET_CALIBRATION_COEFFICIENT -- for polynomial calibration
        (in  ITEM:               ENDITEM_WITH_RAW_VALUE;
         in  COEFFICIENT_NUMBER: COEFFICIENT_RANGE;
         out COEFFICIENT_VALUE:  LONG_REAL;
         out STATUS:             UCL_RETURN);

   -- Description
   --   Gets one polynomial coefficient for the calibration definition of an
   --   enditem.
   -- Parameters:
   --   IN:
   --     ITEM:               enditem to be changed the calibration description
   --     COEFFICIENT_NUMBER: Number of the coefficient
   --   OUT:
```

Copyright per DIN 34

```
--     COEFFICIENT_VALUE: value of the coefficient
--     STATUS:            returns the UCL Return Code values as defined in GROUND_COMMON:
--      OK
--      CAL_ERROR    : An exception occured when getting the new coefficient
--      ITEM_UNKNOWN : Enditem is not known on the executing node


4:  guarded procedure SET_IDENTICAL_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_RAW_VALUE;
       out STATUS : UCL_RETURN);

-- Description
--     Sets the calibration definition of an enditem to "IDENTICAL"
-- Parameters:
--   IN:
--     ITEM:               enditem to be changed the calibration description
--   OUT:
--     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
--      OK                  : Identical Calibration applied
--      ACQUISITION_ENABLED : calibration cannot be changed if acquisition of item is already
--      CAL_ERROR           : An exception occured when setting the new numberr
--      ITEM_UNKNOWN        : Enditem is not known on the executing node


5:  guarded procedure SET_POINT_PAIR_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_RAW_VALUE;
       in  PAIRS  : array of POINT_PAIRS;
       in  LAST   : INTEGER := 0;
       out STATUS : UCL_RETURN);

-- Description
--     Sets a new point pair calibration definition of an enditem
--     Type of calibration is automatically changed to "POINT_PAIRS"
-- Parameters:
--   IN:
--     ITEM:               enditem to be changed the calibration description
--     PAIRS  defines the array of POINT_PAIRS
--     LAST   defines the last valid index of the array, will be ignored if out of bounds
--   OUT:
--     LAST defines the last valid index of the array, will be ignored if out of bounds
--     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
--      OK                  : New point pairs applied
--      ACQUISITION_ENABLED : calibration cannot be changed if acquisition of item is already
--      CAL_ERROR           : An exception occured when setting the point pairs
--      ITEM_UNKNOWN        : Enditem is not known on the executing node


6:  procedure GET_POINT_PAIR_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_RAW_VALUE;
       out PAIRS  : array of POINT_PAIRS;
       out LAST   : INTEGER;
       out STATUS : UCL_RETURN);
-- Description
--     returns the point pair calibration definition of an enditem
-- Parameters:
--   IN:
--     ITEM:               enditem to be changed the calibration description
--   OUT:
--     PAIRS  defined array of POINT_PAIRS
--     LAST   is the last valid index of the defined calibration
--            LAST may be greather than the range of PAIRS, this means
--            the returned array of STATECODE_PAIRS is incomplete!
--     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
--      OK                  : operation successful
--      ITEM_UNKNOWN        : Enditem is not known on the executing node
--      INVALID_SIZE        : PAIRS array is to small (LAST > High (PAIRS))


7:  guarded procedure SET_STATE_CODE_CALIBRATION
      (in  ITEM   : DISCRETE_VALUE;
       in  OTHER  : STATECODE;
       in  PAIRS  : array of STATECODE_PAIRS;
       in  LAST   : INTEGER := 0;
       out STATUS : UCL_RETURN);

-- Description
--     Sets a new statecode calibration definition of an enditem
-- Parameters:
--   IN:
--     OTHER  defines the the statecode for undefined calibration
--     PAIRS  defines the array of STATECODE_PAIRS
--            One STATECODE_PAIR consist of three mandatory elements:
--            RAW_VALUE, HIGH_RAW_VALUE and ENGINEERING_VALUE
--            If the RAW_VALUE of the STATECODE_PAIR is equal to the HIGH_RAW_VALUE
--            a single calibration is defined, otherwise a range calibration is defined.
--            The RAW_VALUE should be less or equal than the HIGH_RAW_VALUE.
--     LAST   defines the last valid index of the array, will be ignored if out of bounds
--   OUT:
--     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
--      OK                  : New discrete calibration pair applied
--      ACQUISITION_ENABLED : calibration cannot be changed if acquisition of item is already
```

Copyright per DIN 34

```
--      CAL_ERROR           : An exception occured when setting the discrete calibration pair
--      ITEM_UNKNOWN        : Enditem is not known on the executing node


 8:  procedure GET_STATE_CODE_CALIBRATION
       (in  ITEM   : DISCRETE_VALUE;
        out OTHER  : STATECODE;
        out PAIRS  : array of STATECODE_PAIRS;
        out LAST   : INTEGER;
        out STATUS : UCL_RETURN);

 -- Description
 --     returns the state code calibration definition of an enditem
 -- Parameters:
 --   OUT:
 --     OTHER  defines the the statecode for undefined calibration
 --     PAIRS  defines the array of STATECODE_PAIRS
 --            If the RAW_VALUE of the STATECODE_PAIR is equal to the
 --            HIGH_RAW_VALUE a single calibration is defined, otherwise
 --            a range calibration is defined.
 --     LAST   is the last valid index of the defined statecode calibration
 --            LAST may be greather than the range of PAIRS, this means
 --            the returned array of STATECODE_PAIRS is incomplete!
 --     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
 --      OK                 : operation successful
 --      ITEM_UNKNOWN       : Enditem is not known on the executing node
 --      INVALID_SIZE       : PAIRS array is to small (LAST > High (PAIRS))


 9:  guarded procedure SET_INTEGER_STRING_CALIBRATION
       (in  ITEM   : ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
        in  OTHER  : STRING;
        in  PAIRS  : array of INTEGER_STRING_PAIRS;
        in  LAST   : INTEGER := 0;
        out STATUS : UCL_RETURN);

 -- Description
 -- see above (SET_STATE_CODE_CALIBRATION)

10:  procedure GET_INTEGER_STRING_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       out OTHER  : STRING;
       out PAIRS  : array of INTEGER_STRING_PAIRS;
       out LAST   : INTEGER;
       out STATUS : UCL_RETURN);

 -- Description
 --     returns the integer – string calibration definition of an enditem
 -- Parameters:
 --   OUT:
 --     OTHER  defines the the statecode for undefined calibration
 --     PAIRS  defines the array of INTEGER_STRING_PAIRS
 --            If the RAW_VALUE of the INTEGER_STRING_PAIRS is equal to the
 --            HIGH_RAW_VALUE a single calibration is defined, otherwise a
 --            range calibration is defined.
 --     LAST   is the last valid index of the defined statecode calibration
 --            LAST may be greather than the range of PAIRS, this means
 --            the returned array of INTEGER_STRING_PAIRS is incomplete!
 --     STATUS returns the UCL Return Code values as defined in GROUND_COMMON
 --      OK                 : operation successful
 --      ITEM_UNKNOWN       : Enditem is not known on the executing node
 --      INVALID_SIZE       : PAIRS array is to small (LAST > High (PAIRS))
 --                           or
 --                           parameter OTHER is to small

11:  guarded procedure SET_UNSIGNED_STRING_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       in  OTHER  : STRING;
       in  PAIRS  : array of UNSIGNED_STRING_PAIRS;
       in  LAST   : INTEGER := 0;
       out STATUS : UCL_RETURN);

 -- Description
 -- see above (SET_STATE_CODE_CALIBRATION)

12:  procedure GET_UNSIGNED_STRING_CALIBRATION
      (in  ITEM   : ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       out OTHER  : STRING;
       out PAIRS  : array of UNSIGNED_STRING_PAIRS;
       out LAST   : INTEGER;
       out STATUS : UCL_RETURN);

 -- Description
 -- see above (GET_INTEGER_STRING_CALIBRATION)

 --------------------------
 -- statecode
 --------------------------

13:  procedure GET_CALIBRATED_STATECODE_FOR_UNSIGNED
      (in  ITEM:      DISCRETE_ENDITEM;
```

Copyright per DIN 34

```
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: STATECODE;
       out STATUS:    UCL_RETURN);

 -- Description:
 --   Returns the engineering value of an enditem by applying the calibration
 --   description. (derived from the calibration attributes of the measurement)
 -- Parameters:
 --   IN:
 --     ITEM:          enditem with calibration description to be used
 --     RAW_VALUE:     value to calibrate
 --   OUT:
 --     ENG_VALUE:     the engineering value of the enditem.
 --     STATUS:        returns the UCL Return Code values
 --       OK:           Decalibration successful
 --       CAL_ERROR:    Decalibration Failed
 --       ITEM_UNKNOWN: Enditem is not known on the executing node

 --------------------------
 -- byte stream
 --------------------------

14:  procedure GET_CALIBRATED_BYTE_STREAM_FOR_UNSIGNED
      (in  ITEM:      ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: STRING;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

15:  procedure GET_CALIBRATED_BYTE_STREAM_FOR_INTEGER
      (in  ITEM:      ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       in  RAW_VALUE: INTEGER;
       out ENG_VALUE: STRING;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

 --------------------------
 -- real
 --------------------------

16:  procedure GET_CALIBRATED_REAL_FOR_INTEGER
      (in  ITEM:      FLOAT_ENDITEM;
       in  RAW_VALUE: INTEGER;
       out ENG_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

17:  procedure GET_CALIBRATED_REAL_FOR_UNSIGNED
      (in  ITEM:      FLOAT_ENDITEM;
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

18:    procedure GET_CALIBRATED_REAL_FOR_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  RAW_VALUE: REAL;
       out ENG_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

19:    procedure GET_CALIBRATED_REAL_FOR_LONG_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  RAW_VALUE: LONG_REAL;
       out ENG_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

 --------------------------
 -- long real
 --------------------------

20:  procedure GET_CALIBRATED_LONG_REAL_FOR_INTEGER
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  RAW_VALUE: INTEGER;
       out ENG_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
```

Copyright per DIN 34

```
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

21:  procedure GET_CALIBRATED_LONG_REAL_FOR_UNSIGNED
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

22:  procedure GET_CALIBRATED_LONG_REAL_FOR_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  RAW_VALUE: REAL;
       out ENG_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

23:  procedure GET_CALIBRATED_LONG_REAL_FOR_LONG_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  RAW_VALUE: LONG_REAL;
       out ENG_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

 --------------------------
 -- integer
 --------------------------

24:  procedure GET_CALIBRATED_INTEGER_FOR_INTEGER
      (in  ITEM:      INTEGER_ENDITEM;
       in  RAW_VALUE: INTEGER;
       out ENG_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

25:  procedure GET_CALIBRATED_INTEGER_FOR_UNSIGNED
      (in  ITEM:      INTEGER_ENDITEM;
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

26:  procedure GET_CALIBRATED_INTEGER_FOR_REAL
      (in  ITEM:      INTEGER_ENDITEM;
       in  RAW_VALUE: REAL;
       out ENG_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

27:  procedure GET_CALIBRATED_INTEGER_FOR_LONG_REAL
      (in  ITEM:      INTEGER_ENDITEM;
       in  RAW_VALUE: LONG_REAL;
       out ENG_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

 --------------------------
 -- unsigned integer
 --------------------------

28:  procedure GET_CALIBRATED_UNSIGNED_FOR_INTEGER
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  RAW_VALUE: INTEGER;
       out ENG_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

29:  procedure GET_CALIBRATED_UNSIGNED_FOR_UNSIGNED
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  RAW_VALUE: UNSIGNED_INTEGER;
       out ENG_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)
```

Copyright per DIN 34

```
30:  procedure GET_CALIBRATED_UNSIGNED_FOR_REAL
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  RAW_VALUE: REAL;
       out ENG_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

31:  procedure GET_CALIBRATED_UNSIGNED_FOR_LONG_REAL
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  RAW_VALUE: LONG_REAL;
       out ENG_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_CALIBRATED_STATECODE_FOR_UNSIGNED)

 ----------------------------
 --  Decalibration Functions
 ----------------------------

 --------------------------
 -- statecode
 --------------------------

32:  procedure GET_DECALIBRATED_VALUE_FOR_STATECODE
      (in  ITEM:      DISCRETE_ENDITEM;
       in  ENG_VALUE: STATECODE;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 --   Returns the raw value of an discrete enditem by applying the
 --   decalibration (derived from the calibration attributes of the measurement
 --   resp. decalibration attributes of the stimulus parameter)
 --      Measurement: CALIBRATION info –> convert –> apply
 --      Stimulus: DECALIBRATION info –> apply
 -- Parameters:
 --   IN:
 --     ITEM:          enditem with calibration description to be used
 --     ENG_VALUE:     value to calibrate
 --   OUT:
 --     RAW_VALUE:     the raw value of the enditem.
 --     STATUS:        returns the UCL Return Code values
 --       OK          : Decalibration successful
 --       CAL_ERROR   : Decalibration Failed: Statecode not defined in
 --                     Discrete Calibration
 --       ITEM_UNKNOWN: Enditem is not known on the executing node

 --------------------------
 -- byte stream
 --------------------------

33:  procedure GET_DECALIBRATED_UNSIGNED_VALUE_FOR_BYTE_STREAM
      (in  ITEM:      ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       in  ENG_VALUE: STRING;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_VALUE_FOR_STATECODE)

34:  procedure GET_DECALIBRATED_INTEGER_VALUE_FOR_BYTE_STREAM
      (in  ITEM:      ENDITEM_WITH_BYTESTREAM_RAW_VALUE;
       in  ENG_VALUE: STRING;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_VALUE_FOR_STATECODE)

 --------------------------
 -- real
 --------------------------

35:  procedure GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  ENG_VALUE: REAL;
       out RAW_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 --   Returns the raw value of an enditem by applying the decalibration
 --   description. (derived from the calibration attributes of the measurement
 --   resp. decalibration attributes of the stimulus parameter)
 --
 --   Note: Measurements bear a calibration function, while stimuli (i.e.
 --     their parameter)  bear a decalibration function. Thus for  measurements,
 --     the function specified in the MDB is converted to a decalibration
 --     function and then applied. For stimuli, the function specified in the
```

Copyright per DIN 34

```
--     MDB is used to calculate RAW_VALUE from the ENG_VALUE
--
--   Measurement: CALIBRATION   info -> convert -> apply
--   Stimulus:    DECALIBRATION info -> apply
--
-- Parameters:
--   IN:
--     ITEM:          enditem with calibration description to be used
--     ENG_VALUE:     value to calibrate
--   OUT:
--     RAW_VALUE:     the raw value of the enditem.
--     STATUS:        returns the UCL Return Code values
--       OK:           Decalibration successful
--       CAL_ERROR:    Decalibration Failed
--       ITEM_UNKNOWN: Enditem is not known on the executing node

36:  procedure GET_DECALIBRATED_UNSIGNED_VALUE_FOR_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  ENG_VALUE: REAL;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

37:  procedure GET_DECALIBRATED_REAL_VALUE_FOR_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  ENG_VALUE: REAL;
       out RAW_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

38:  procedure GET_DECALIBRATED_LONG_REAL_VALUE_FOR_REAL
      (in  ITEM:      FLOAT_ENDITEM;
       in  ENG_VALUE: REAL;
       out RAW_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

 --------------------------
 -- long real
 --------------------------

39:  procedure GET_DECALIBRATED_INTEGER_VALUE_FOR_LONG_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  ENG_VALUE: LONG_REAL;
       out RAW_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

40:  procedure GET_DECALIBRATED_UNSIGNED_VALUE_FOR_LONG_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  ENG_VALUE: LONG_REAL;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

41:  procedure GET_DECALIBRATED_REAL_VALUE_FOR_LONG_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  ENG_VALUE: LONG_REAL;
       out RAW_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

42:  procedure GET_DECALIBRATED_LONG_REAL_VALUE_FOR_LONG_REAL
      (in  ITEM:      DOUBLE_FLOAT_ENDITEM;
       in  ENG_VALUE: LONG_REAL;
       out RAW_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

 --------------------------
 -- integer
 --------------------------

43:  procedure GET_DECALIBRATED_INTEGER_VALUE_FOR_INTEGER
      (in  ITEM:      INTEGER_ENDITEM;
       in  ENG_VALUE: INTEGER;
       out RAW_VALUE: INTEGER;
```

Copyright per DIN 34

```
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)


44:  procedure GET_DECALIBRATED_UNSIGNED_VALUE_FOR_INTEGER
      (in  ITEM:      INTEGER_ENDITEM;
       in  ENG_VALUE: INTEGER;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)


45:  procedure GET_DECALIBRATED_REAL_VALUE_FOR_INTEGER
      (in  ITEM:      INTEGER_ENDITEM;
       in  ENG_VALUE: INTEGER;
       out RAW_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)


46:  procedure GET_DECALIBRATED_LONG_REAL_VALUE_FOR_INTEGER
      (in  ITEM:      INTEGER_ENDITEM;
       in  ENG_VALUE: INTEGER;
       out RAW_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

 --------------------------
 -- unsigned integer
 --------------------------

47:  procedure GET_DECALIBRATED_INTEGER_VALUE_FOR_UNSIGNED
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  ENG_VALUE: UNSIGNED_INTEGER;
       out RAW_VALUE: INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

48:  procedure GET_DECALIBRATED_UNSIGNED_VALUE_FOR_UNSIGNED
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  ENG_VALUE: UNSIGNED_INTEGER;
       out RAW_VALUE: UNSIGNED_INTEGER;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

49:  procedure GET_DECALIBRATED_REAL_VALUE_FOR_UNSIGNED
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  ENG_VALUE: UNSIGNED_INTEGER;
       out RAW_VALUE: REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

50:  procedure GET_DECALIBRATED_LONG_REAL_VALUE_FOR_UNSIGNED
      (in  ITEM:      UNSIGNED_INTEGER_ENDITEM;
       in  ENG_VALUE: UNSIGNED_INTEGER;
       out RAW_VALUE: LONG_REAL;
       out STATUS:    UCL_RETURN);

 -- Description:
 -- see above (GET_DECALIBRATED_INTEGER_VALUE_FOR_REAL)

end CALIBRATION_LIB;
```

Copyright per DIN 34

I-17 HLCL Ground System library Specification: HLCL_CSS_CMDS

The library HLCL_CSS_CMDS contains procedures to define CSS specific prededfined HLCL commands.

I-17.1 HLCL System library Specification

```
------------------------------------------------------------------------
--
-- **************************
--  System UCL System Library
-- **************************
--
-- ABSTRACT
--     Defines CSS specific HLCL commands.
--     This library is for use in HLCL procedures only.
--
--     Must be compiled with Body Id = 0
--                       for Language = HLCL.
--
--     recommended Nickname: ”HLCL_CSS_CMDS”
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : System System-Library
--      CGS CM       : ”$Id: //cgs/7.3.DEV/src/distribution/gsaf/cgsi/lib/ucl/hlcl_css_cmds_.ucl#5 $”
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library HLCL_CSS_CMDS;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type ACTIVATION = (STEPWISE, CONTINUOUS);

   type CSS_FB_DATABASE_ITEMS = pathname (TOPLEVEL_COMPOSITE_FB,
                                          COMPOSITE_FB,
                                          CONSTANT_FB,
                                          ASYNCHRONOUS_FB,
                                          SYNCHRONOUS_FB);

   type CSS_FB_DATABASE_ITEMS_WITH_DEFAULT = pathname (TOPLEVEL_COMPOSITE_FB,
                                                       COMPOSITE_FB,
                                                       CONSTANT_FB,
                                                       ASYNCHRONOUS_FB,
                                                       SYNCHRONOUS_FB,
                                                       VIRTUAL);

   type DESTINATION = (LOGFILE, SCREEN);

   type DISABLE_MODE = (CALCULATE, FROZEN);

   type LOGGING_ATTRIBUTE = (CYCLIC, ON_CHANGE);

   type LOGGING_DESTINATION = (ARCHIVE_ONLY, ARCHIVE_AND_DATA_SET);

   type PRIVILEGE = (LOG_PRIVILEGE, WRITE_PRIVILEGE, START_PRIVILEGE);
   -- LOG_PRIVILEGE allows to log, snapshot and trace items in log file
   -- WRITE_PRIVILEGE allows to assign values to model items
   -- START_PRIVILEGE (i.e. session ownership) allows to
   --   * start, stop, and abort the simulation
   --   * load and store simulation state (state vector)
   --   * set minframe interval/increment
   --   * set simulated mission time
   --   * grant and withdraw privileges to/from other users
   -- START_PRIVILEGE it is initially assigned to the user who started CSS_UI
   -- here users are identified by combination of user name and host name

   type SIMULATOR_MODE = (REALTIME, STANDALONE);

   type SIMULATION_STEP = UNSIGNED_INTEGER (0 .. 4294967295);

   type TABLE_ACTION = (ADD, REMOVE);

   type TABLE_CLASS = (MONITORING, LOGGING);

   type TIME_CLASS = (LOCAL, SMT);

   type TIME_MODE = (VIRTUAL, AUTO);
```

Copyright per DIN 34

```
------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  ----------------------
  -- Command execution --
  ----------------------

  -- ---------------------------------------------------------------------------
  --
  -- ---------------------------------------------------------------------------

  procedure START_CSS_UI
    (in MODEL : CSS_FB_DATABASE_ITEMS;
     in HOST  : STRING := "";
     in MODE  : SIMULATOR_MODE := REALTIME);

  -- Description:
  --   Start the CSS user inteface.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --     HOST  : name of target machine on which to start CSS UI
  --             by default name of local machine running ICP
  --     MODE  : simulation mode (REALTIME or STANDALONE)
  -- Constraints:
  --   at maximum 10 CSS_UI instances can be started in parallel from an ICP instance
  --   not more than 1 CSS_UI/simulator instance per simulation model at a time can be started from an ICP instance
  -- Note:
  --   simulation mode REALTIME is more restrictive
  --   * simulation attributes are fixed (i.e. time mode VIRTUAL, factor 1.0 (minframe duration 200 ms), activation mode CONTINUOUS)
  --   * basic simulation control (i.e. start/stop) is done by HLCL commands START_SMT and STOP_SMT
  --   * various CSS specific HLCL commands are disabled
  -- Example:
  --   START_CSS_UI \EURECA\SIMULATOR\TEST\MODEL_1, "ws_node_1"

  procedure STOP_CSS_UI
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Stop the CSS user interface.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI instance to be affected
  --             can be omitted if only one CSS_UI instance started from ICP instance
  -- Constraints:
  --   CSS_UI on given simulation model started

  procedure IS_CSS_UI_STARTED
    (in  MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \;
     out RESULT : BOOLEAN);

  -- Description:
  --   Check whether CSS_UI has been started on given simulation model.
  -- Parameters:
  --   IN:
  --     MODEL  : pathname of simulation model
  --              can be omitted if only one, if any, CSS_UI instance started from ICP instance
  --   OUT:
  --     RESULT : true if CSS_UI started, otherwise false
  -- Constraints:
  --   none

  -- ---------------------------------------------------------------------------
  --
  -- ---------------------------------------------------------------------------

  procedure SET_SIMULATION_ATTRIBUTES
    (in ACTIVATION : ACTIVATION;
     in STEPS      : SIMULATION_STEP;
     in TIME_MODE  : TIME_MODE;
     in FACTOR     : REAL := 1.0;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Set simulation attributes applicable to starting simulator kernel (via START_CSS_KERNEL).
  -- Parameters:
  --   IN:
  --     ACTIVATION : activation mode (STEPWISE, CONTINUOUS)
  --                  in STEPWISE mode the simulation is suspended automatically after given number of simulation steps (minframes) have
been executed
  --                  in CONTINUOUS mode the simulation proceeds until suspended by STOP_SIMULATION
  --     STEPS      : number of simulation steps (i.e. minframes) to be executed on START_SIMULATION if activation mode is STEPWISE
  --     TIME_MODE  : time mode (AUTOMATIC, VIRTUAL)
  --                  in AUTOMATIC mode the simulation is executed as fast as possible, next simulation step is started as soon as process-
ing of previous step has finished (dynamic frame durations in LOT)
  --                  in VIRTUAL mode ths simulation is executed with fixed frame durations in LOT (static time raster)
  --                  after processing of simulation step has finished the simulation pauses until minframe duration expired
  --     FACTOR     : minframe duration in LOT given as multiple of 200 ms (only in time mode VIRTUAL)
  --                  minframe duration has impact on duration of higher order synchronous frames given as multiples of the minframe
```

Copyright per DIN 34

```
--                  might want to adapt also minframe increment (see SET_MINFRAME)
--     MODEL      : pathname of simulation model
--                  denotes CSS_UI instance to be affected
--                  can be omitted if only one CSS_UI instance started from ICP instance
-- Constraints:
--   CSS_UI on given simulation model started in simulation mode STANDALONE
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
--   simulation state loaded
--   simulation suspended
--   minimum factor 0.25 (minimum minframe duration 50 ms)
--   minimum steps 1
--   in virtual mode the simulation model implementation must allow processing of all frames in time in worst case

procedure SET_MINFRAME
  (in DURATION : DURATION;
   in MODEL    : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Set minframe increment, i.e. the amount the SMT is incremented per simulation step (minframe).
-- Parameters:
--   IN:
--     DURATION : SMT increment
--     MODEL    : pathname of simulation model
--                denotes CSS_UI instance to be affected
--                can be omitted if only one CSS_UI instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started in simulation mode STANDALONE
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
--   simulation state not yet loaded or simulation initially suspended
-- Note:
--   the minframe increment does not affect the minframe duration in LOT
-- Example:
--   SET_MINFRAME 5.0 [s]

procedure SET_TIME
  (in TIME  : TIME;
   in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Set initial value for SMT.
-- Parameters:
--   IN:
--     TIME  : initial value for SMT
--     MODEL : pathname of simulation model
--             denotes CSS_UI instance to be affected
--             can be omitted if only one CSS_UI instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started in simulation mode STANDALONE
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
--   simulation state not yet loaded or simulation initially suspended
-- Example:
--   SET_TIME 24.12.1996 13:27:55.50

-- ------------------------------------------------------------------------------
--
-- ------------------------------------------------------------------------------

procedure START_CSS_KERNEL
  (in KERNEL_HOST : STRING;
   in CMAS_HOST   : STRING;
   in MODEL       : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Start simulator kernel (and possibly CMAS) on given host(s).
-- Parameters:
--   IN:
--     KERNEL_HOST : name of host to start simulator kernel on
--     CMAS_HOST   : name of host to start CMAS on
--                   empty string if kernel should be started without CMAS
--     MODEL       : pathname of simulation model
--                   denotes CSS_UI/simulator instance to be affected
--                   can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI on given simulation model started
--   simulator kernel not yet connected
-- Example:
--   START_CSS_KERNEL "sim_node_1", ""

procedure STOP_CSS_KERNEL
  (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Stop simulator kernel.
-- Parameters:
--   IN:
--     MODEL : pathname of simulation model
--             denotes CSS_UI/simulator instance to be affected
--             can be omitted if only one CSS_UI/simulator instance started from ICP instance
```

Copyright per DIN 34

```
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
--   simulation state not yet loaded or simulation suspended or simulation faulty halted

procedure CONNECT_TO_CSS_KERNEL
  (in HOST  : STRING := "";
   in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Connect to a running simulator kernel.
-- Parameters:
--   IN:
--     HOST  : name of host running simulator kernel
--             by default local host running ICP
--     MODEL : pathname of simulation model
--             denotes CSS_UI/simulator instance to be affected
--             can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   simulator kernel running but not yet connected

procedure DISCONNECT_FROM_CSS_KERNEL
  (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Disconnect from a running simulator kernel (leaving
--   it up and running).
-- Parameters:
--   IN:
--     MODEL : pathname of simulation model
--             denotes CSS_UI/simulator instance to be affected
--             can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   simulator kernel running and connected

procedure IS_CONNECTED_TO_KERNEL
  (in  HOST   : STRING := "";
   in  MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \;
   out RESULT : BOOLEAN);

-- Description:
--   Check whether CSS_UI is connected to simulator kernel on given host.
-- Parameters:
--   IN:
--     HOST   : name of host running simulator kernel
--     MODEL  : pathname of simulation model
--              denotes CSS_UI/simulator instance to be affected
--              can be omitted if only one CSS_UI/simulator instance started from ICP instance
--   OUT:
--     RESULT : true if connected, otherwise false
-- Constraints:
--   none

-- ------------------------------------------------------------------------
--
-- ------------------------------------------------------------------------

procedure LOAD
  (in VECTOR : STRING;
   in MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Load simulation state (state vector).
-- Parameters:
--   IN:
--     VECTOR : name of simulation state (state vector) to be loaded
--     MODEL  : pathname of simulation model
--              denotes CSS_UI/simulator instance to be affected
--              can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
--   simulation state not yet loaded or simulation suspended
--   given simulation state (state vector) existing

procedure STORE
  (in VECTOR : STRING;
   in MODEL  : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Store simulation state (state vector).
-- Parameters:
--   IN:
--     VECTOR : name of simulation state (state vector)
--     MODEL  : pathname of simulation model
--              denotes CSS_UI/simulator instance to be affected
--              can be omitted if only one CSS_UI/simulator instance started from ICP instance
```

Copyright per DIN 34

```
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation suspended or faulty halted
  --   name must comply with Ada identifier syntax, at maximum 16 characters

  -- ---------------------------------------------------------------------------
  --
  -- ---------------------------------------------------------------------------

  procedure START_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Start or continue a simulation.
  --   Behaviour depends on current simulation attributes as set by SET_SIMULATION_ATTRIBUTES
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --             can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started in simulation mode STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation suspended
  -- Note:
  --   in simulation mode REALTIME the simulation is controlled by HLCL commands START_SMT and STOP_SMT.

  procedure STOP_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Suspend a running simulation, halt SMT.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --             can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started in simulation mode STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation running

  procedure ABORT_SIMULATION
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Abort simulation in faulty state.
  --   After aborting a simulation the user can only store a state vector or shut-down the simulator.
  -- Parameters:
  --   IN:
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                  can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started in simulation mode STANDALONE
  --   user needs START_PRIVILEGE (session ownership)
  --   simulator kernel running and connected
  --   simulation state loaded
  --   simulation running faulty

  -- ---------------------------------------------------------------------------
  --
  -- ---------------------------------------------------------------------------

  procedure SET_ASSIGN
    (in DISABLE : DISABLE_MODE := CALCULATE;
     in TIME    : union (TIME, DURATION) := 0.0 [s];
     in CLASS   : TIME_CLASS := SMT);

  -- Description:
  --   Set attributes applicable to assignments to simulation model subitems.
  -- Parameters:
  --   IN:
  --     DISABLE : configure assignments to be persistent (perform implicit FREEZE) or not
  --               by default assignments will not be persistent (i.e. assigned value will be overridden by next assignment in function
block implementation when executed)
  --     TIME    : configure execution of assignments
  --               immediate execution   : 0.0 [s]
  --               time tagged execution : either at fixed time or with delay relative to time of assignment
  --               by default assignments will be executed immediately
  --     CLASS   : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   none
  -- Example:
```

Copyright per DIN 34

```
  --   SET_ASSIGN CALCULATE, 100.0, LOCAL
  --   \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.OUTPUT = 305.76
  --   -- value of variable BLOCK_A.OUTPUT will be set to 305.76 with delay of 100 sec in local time
  --   SET_ASSIGN FROZEN
  --   \EURECA\TEST\MODEL_1\BOOM1\VALVE3.OUTPUT = 0.0
  --   -- value of variable VALVE3.OUTPUT will be set to 0.0 and frozen immediately, i.e. value will stay 0.0 until released via REACTIVATE

  procedure FREEZE
    (in SUBITEM_PATH : pathname.*;
     in TIME         : union (TIME, DURATION) := 0.0 [s];
     in CLASS        : TIME_CLASS := SMT);

  -- Description:
  --   Freeze value of simulation model subitem.
  --   Lock subitem by disabling updates from
  --     * internal: assignments in simulation model implementation (atomic function blocks)
  --     * external: assignments via CSS_UI or HLCL and updates from CMAS, if connected
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     TIME         : immediate execution   : 0.0 [s]
  --                    time tagged execution : either at fixed time or with delay relative to current time
  --                    by default the command will be executed immediately
  --     CLASS        : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs WRITE_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input, atomic function block output, parameter output, composite function block output but
not of pulse or burst pulse type

  procedure REACTIVATE
    (in SUBITEM_PATH : pathname.*;
     in TIME         : union (TIME, DURATION) := 0.0 [s];
     in CLASS        : TIME_CLASS             := SMT);

  -- Description:
  --   Revert effects of FREEZE command.
  --   Release lock on simulation model subitem by enabling updates from
  --     * internal: assignments in simulation model implementation (atomic function blocks)
  --     * external: assignments via CSS_UI or HLCL and updates from CMAS, if connected
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     TIME         : immediate execution   : 0.0 [s]
  --                    time tagged execution : either at fixed time or with delay relative to current time
  --                    by default the command will be executed immediately
  --     CLASS        : reference system for time (LOCAL, i.e. LOT or SMT)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs WRITE_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input, atomic function block output, parameter output, composite function block output but
not of pulse or burst pulse type

  procedure CANCEL_PENDING_COMMANDS
    (in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Cancel all pending (outstanding time tagged) commands issued by user.
  -- Parameters:
  --   IN:
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --             can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  --   there must be pending commands, all pending commands must be cancelable

  -- ------------------------------------------------------------------------
  --
  -- ------------------------------------------------------------------------

  procedure ACTIVATE
    (in DEFINITION : TABLE_CLASS := LOGGING;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Enable monitoring or logging.
  -- Parameters:
  --   IN:
  --     DEFINITION : kind of operation to be enabled (MONITORING, LOGGING)
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                  can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
```

Copyright per DIN 34

```
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if definition is LOGGING
  --   simulator kernel running and connected
  --   simulation state loaded

  procedure DEACTIVATE
    (in DEFINITION : TABLE_CLASS := LOGGING;
     in MODEL      : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Disable monitoring or logging.
  -- Parameters:
  --   IN:
  --     DEFINITION : kind of operation to be disabled (MONITORING, LOGGING)
  --     MODEL      : pathname of simulation model
  --                  denotes CSS_UI/simulator instance to be affected
  --                  can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if definition is LOGGING
  --   simulator kernel running and connected
  --   simulation state loaded

  procedure LOAD_TABLE
    (in TABLE : STRING;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Load given simulation table containing monitoring, logging and/or tracing definitions.
  -- Parameters:
  --   IN:
  --     TABLE : name of simulation table
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --             can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table existing
  --   (name must comply with Ada identifier syntax, at maximum 16 characters)

  procedure UNMONITOR
    (in TABLE     : STRING;
     in ITEM_PATH : pathname.*);

  -- Description:
  --   Remove monitoring elements on given simulation model subitem from given simulation table by removing monitoring definition from simu-
lation table.
  -- Parameters:
  --   IN:
  --     TABLE     : name of simulation table to be updated
  --     ITEM_PATH : pathname of simulation model subitem
  --                 implicitely denotes CSS_UI/simulator instance to be affected
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --   subitem one of simulation model top level input/output, atomic function block input/output, parameter output, composite function
block output already monitored in given table

  procedure LOG_ITEM
    (in ACTION       : TABLE_ACTION;
     in TABLE        : STRING;
     in SUBITEM_PATH : pathname.*;
     in ATTRIBUTE    : LOGGING_ATTRIBUTE   := CYCLIC;
     in DESTINATION  : LOGGING_DESTINATION := ARCHIVE_ONLY);

  -- Description:
  --   Register/unregister given simulation model subitem to be logged by adding/removing logging definition to/from given simulation table.
  -- Parameters:
  --   IN:
  --     ACTION       : kind of action to be performed (ADD or REMOVE)
  --     TABLE        : name of simulation table to be updated
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     ATTRIBUTE    : logging mode (CYCLIC, ON_CHANGE)
  --     DESTINATION  : logging destination(s) (ARCHIVE_ONLY, ARCHIVE_AND_DATA_SET)
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs LOG_PRIVILEGE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --   if action = ADD    : subitem one of simulation model top level input/output, atomic function block input/output, parameter output,
composite function block output
  --                        equivalent subitem not already logged
  --                        subitem not of VECTOR, MATRIX, COMPLEX, DURATION or TIME type if destination = ARCHIVE_AND_DATA_SET
  --   if action = REMOVE : equivalent subitem already logged in given table
```

Copyright per DIN 34

```
  procedure TRACE
    (in ACTION      : TABLE_ACTION;
     in TABLE       : STRING;
     in BLOCK_PATH  : CSS_FB_DATABASE_ITEMS;
     in DESTINATION : DESTINATION := SCREEN);

  -- Description:
  --   Register/unregister given atomic function block to be traced (execution tracing) by adding/removing tracing definition to/from given
simulation table.
  -- Parameters:
  --   IN:
  --     ACTION      : kind of action to be performed (ADD or REMOVE)
  --     TABLE       : name of simulation table to be updated
  --     BLOCK_PATH  : pathname of atomic function block
  --                   implicitely denotes CSS_UI/simulator instance to be affected
  --     DESTINATION : output destination, i.e. SCREEN (CSS_UI console window) or LOGFILE
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   user needs LOG_PRIVILEGE if destination is LOGFILE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   given simulation table loaded
  --   if action = ADD    : atomic function block not already traced to destination
  --   if action = REMOVE : atomic function block already traced to destination in given simulation table

  procedure SNAPSHOT
    (in SUBITEM_PATH : pathname.*;
     in DESTINATION  : DESTINATION := SCREEN);

  -- Description:
  --   Write current model item value once into logfile or onto screen (CSS_UI console window).
  -- Parameters:
  --   IN:
  --     SUBITEM_PATH : pathname of simulation model subitem
  --                    implicitely denotes CSS_UI/simulator instance to be affected
  --     DESTINATION  : output destination, i.e. SCREEN (CSS_UI console window) or LOGFILE
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   user needs LOG_PRIVILEGE if destination is LOGFILE
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input/output, atomic function block input/output, parameter output, composite function
block output
  -- Example:
  --   SNAPSHOT \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.INPUT LOGFILE

  -- ------------------------------------------------------------------------
  --
  -- ------------------------------------------------------------------------

  procedure WAIT
    (in TIME  : union (TIME, DURATION);
     in CLASS : TIME_CLASS := LOCAL;
     in MODEL : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Suspend command execution until given time or for given duration.
  --   Useful primarily in command sequences.
  -- Parameters:
  --   IN:
  --     TIME  : time to be waited until
  --             duration to be waited for
  --     CLASS : reference system for time (LOCAL, i.e. LOT or SMT)
  --     MODEL : pathname of simulation model
  --             denotes CSS_UI/simulator instance to be affected
  --             can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
  --   simulator kernel running and connected
  -- Example:
  --   WAIT 23.08.1995 07:23:17.000
  --   WAIT 100.0, SMT

  -- ------------------------------------------------------------------------
  --
  -- ------------------------------------------------------------------------

  procedure REQUEST
    (in PRIVILEGE : PRIVILEGE;
     in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

  -- Description:
  --   Request given privilege from session owner.
  -- Parameters:
  --   IN:
  --     PRIVILEGE : privilege requested
  --     MODEL     : pathname of simulation model
  --                 denotes CSS_UI/simulator instance to be affected
  --                 can be omitted if only one CSS_UI/simulator instance started from ICP instance
  -- Constraints:
  --   CSS_UI/simulator on given simulation model started
```

Copyright per DIN 34

```
--   simulator kernel running and connected

procedure GRANT
  (in PRIVILEGE : PRIVILEGE;
   in USER      : STRING;
   in HOST      : STRING;
   in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Grant given privilege to given user on given host.
-- Parameters:
--   IN:
--     PRIVILEGE : privilege to be granted
--     USER      : user name
--     HOST      : name of host running user's CSS_UI
--     MODEL     : pathname of simulation model
--                 denotes CSS_UI/simulator instance to be affected
--                 can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected

procedure WITHDRAW
  (in PRIVILEGE : PRIVILEGE;
   in USER      : STRING;
   in HOST      : STRING;
   in MODEL     : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Withdraw given (previously granted) privilege for given user on given host.
-- Parameters:
--   IN:
--     PRIVILEGE : privilege to be be withdrawn
--     USER      : user name
--     HOST      : name of host running user's CSS_UI
--     MODEL     : pathname of simulation model
--                 denotes CSS_UI/simulator instance to be affected
--                 can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected

procedure BROADCAST
  (in MESSAGE : STRING;
   in USER    : STRING := "*";
   in MODEL   : CSS_FB_DATABASE_ITEMS_WITH_DEFAULT := \);

-- Description:
--   Send message to simulation user(s).
-- Parameters:
--   IN:
--     MESSAGE : message text
--     USER    : name of user to send message to
--               by default message is send to all users
--     MODEL   : pathname of simulation model
--               denotes CSS_UI/simulator instance to be affected
--               can be omitted if only one CSS_UI/simulator instance started from ICP instance
-- Constraints:
--   CSS_UI/simulator on given simulation model started
--   user needs START_PRIVILEGE (session ownership)
--   simulator kernel running and connected
-- Note:

-- ------------------------------------------------------------------------
--
-- ------------------------------------------------------------------------

-- builtin write access to simulation model subitem

-- Description:
--   Assign value to model subitem applying assignment attributes as specified via SET_ASSIGN.
-- Constraints:
--   CSS_UI/simulator on given simulation model (by subitem path) started
--   user needs WRITE_PRIVILEGE
--   simulator kernel running and connected
--   simulation state loaded
--   subitem one of simulation model top level input, atomic function block output, parameter output, composite function block output
--   --if subitem of pulse type value must be TRUE
--   subitems of PULSE resp. BURST_PULSE type must be assigned via TRIGGER command
--   if subitem of burst pulse type value must be > 0
-- Example:
--   \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.UNSIGNED_INTEGER_OUTPUT := 42
--   trigger \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.PULSE_INPUT
--   trigger \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.BURST_PULSE_INPUT, 3

-- builtin read access to simulation model subitem

-- Description:
--   Get value of model subitem.
```

Copyright per DIN 34

```
  -- Constraints:
  --   CSS_UI/simulator on given simulation model (by subitem path) started
  --   simulator kernel running and connected
  --   simulation state loaded
  --   subitem one of simulation model top level input/output, atomic function block input/output, parameter output, composite function
block input/output
  -- Example:
  --   var x : unsigned_integer
  --   x := \APM\SUBSYSTEM_1\PUMP\MODEL_V\COMP_1\BLOCK_A.UNSIGNED_INTEGER_OUTPUT

end HLCL_CSS_CMDS;
```

Copyright per DIN 34

I-18 HLCL Ground System library Specification: HLCL_HCI_CMDS

The library HLCL_HCI_CMDS contains procedures to define HCI specific prededfined HLCL commands.

I-18.1 HLCL System library Specification

```
------------------------------------------------------------------------
--
-- **************************
--  System UCL System Library
-- **************************
--
-- ABSTRACT
--     Defines HCI HLCL specific commands.
--     This library is for use in HLCL procedures only.
--
--     Must be compiled for ground and with Body Id = 0
--
--     recommended Nickname: "HLCL_HCI_CMDS"
--
-- IDENTIFICATION
--      PROJECT NAME : CGS
--      OBJECT NAME  : System System-Library
--      CGS CM       : "$Id: //cgs/MAIN/src/distribution/gsaf/cgsi/lib/ucl/hlcl_hci_cmds_.ucl#2 $"
--*----------------------------------------------------------------------
-- CONTENTS
--        COMPILER   : UCLC
--        LANGUAGE   : UCL
------------------------------------------------------------------------

library HLCL_HCI_CMDS;

------------------------------------------------------------------------
-- TYPES
------------------------------------------------------------------------

   type AVAILABLE_PRINTERS = (LASER_PRINTER1, LASER_PRINTER2);

   type EGSE_NODE     = pathname (EGSE_NODE);

   type PICTURE       = pathname (WDU_GROUND_SYNOPTIC_DISPLAY,
                                  USS_DISPLAY);

   type UCL_AUTOMATED_PROCEDURE
                      = pathname (UCL_AUTOMATED_PROCEDURE);

   type UCL_ITEM_NAME = pathname (EGSE_INTEGER_DERIVED_VALUE,
                                  EGSE_FLOAT_DERIVED_VALUE,
                                  EGSE_DISCRETE_DERIVED_VALUE,
                                  EGSE_STRING_DERIVED_VALUE,
                                  UNSIGNED_INT_DERIVED_VALUE,
                                  DOUBLE_FLOAT_DERIVED_VALUE,
                                  UCL_AUTOMATED_PROCEDURE,
                                  UCL_USER_LIBRARY);

------------------------------------------------------------------------
-- OPERATIONS
------------------------------------------------------------------------

  -----------------------
  -- Command execution --
  -----------------------

  procedure ABORT_AP
    (in AP   : INTEGER;
     in NODE : EGSE_NODE := \\);

  -- Description:
  --   This operation send a command to the given node to abort UCL automated
  --   procedure with given AP identification
  -- Parameters:
  --   IN:
  --     AP         : process number of the AP process
  --     NODE       : Node on which the command is to be executed, by default
  --                  the current node.
  -- Example:
  --     ABORT_AP 3, \NODES\TN_1

  procedure ABORT_ALL_APS
    (in NODE : EGSE_NODE := \\);

  -- Description:
  --   This operation send a command to the given node to abort all UCL
  --   automated procedures.
```

Copyright per DIN 34

```
-- Parameters:
--   IN:
--     NODE       : Node on which the command is to be executed, by default
--                  the current node.

procedure ASSIGN_PICTURE
  (in PICTURE             : PICTURE;
   in WIDTH               : INTEGER := -1;
   in HEIGHT              : INTEGER := -1;
   in HORIZONTAL_POSITION : INTEGER := -1;
   in VERTICAL_POSITION   : INTEGER := -1);

-- Description:
--   Create a synoptic display and assign the picture to it.
-- Parameters:
--   IN:
--     PICTURE             : Name of the picture to be displayed
--     WIDTH               : Width of the synoptic window to be created
--     HEIGHT              : Height of the synoptic window to be created
--     HORIZONTAL_POSITION : X position of the synoptic window to be created
--     VERTICAL_POSITION   : Y position of the synoptic window to be created
-- Example: ASSIGN_PICTURE \DMS_05\PICTURE07, 400, 400, 100, 0

procedure GET_ENVIRONMENT
  (in  NAME  : STRING;
   out VALUE : STRING);

-- Description:
--   Obtain value for environment name
-- Parameters:
--   IN:
--     NAME  : name of the environment variable
--   OUT:
--     VALUE : value of the environment variable
-- Example: variable SAS_HOME: STRING (100);
--          GET_ENVIRONMENT "SAS_HOME", SAS_HOME

procedure HISTORY
  (in  COUNT    : UNSIGNED_INTEGER := 10;
   out COMMANDS : STRING);

-- Description:
--   Display a list of previously entered commands
-- Parameters:
--   IN:
--     COUNT    : number of commands to be displayed
--   OUT:
--     COMMANDS : string of previously entered commands
--                commands are split up by ';'
-- Example: variable CALL_STACK: STRING (1000);
--          HISTORY 20, CALL_STACK

procedure LOAD_UCL
  (in  ITEM   : UCL_ITEM_NAME;
   in  NODE   : EGSE_NODE := \\;
   out STATUS : INTEGER);

-- Description:
--   Loads the UCL I-code of an item into test node.
-- Parameters:
--   IN:
--     ITEM   : item to be loaded
--     NODE   : Node on which the item will be loaded
--   OUT:
--     STATUS : Return status (UCL_RETURN, 1 is success).

procedure PRINT
  (in FILE    : STRING;
   in PRINTER : AVAILABLE_PRINTERS := LASER_PRINTER1);

-- Description:
--   Print a file
-- Parameters:
--   IN:
--     FILE    : file to be printed (including directory path)
--     PRINTER : name of the printer which prints the document


procedure START_HCI_APPLICATION
  (in APPLICATION           : STRING;
   in APPLICATION_PARAMETER : STRING := "";
   in USER_CONFIRMED        : BOOLEAN := TRUE);

-- Description:
--   Start an application program or HCI window application on the workstation
-- Parameters:
--   IN:
--     APPLICATION           : Name of the program. If the application name has
--                             a HCI.prefix the HCI window application is started
--                             (see example 2).
--                             The list of HCI window applications is described
```

Copyright per DIN 34

```
  --                            in CGS user manual 7.3.2.5 Screen Setup Maintenance.
  --     APPLICATION_PARAMETER : If the application needs parameters, they can be
  --                            given here.
  --     USER_CONFIRMED        : If true, the user is ask for confirmation before
  --                            starting the application
  -- Example 1: START_HCI_APPLICATION "${OPENWINHOME}/bin/xterm","",false
  -- Example 2: START_HCI_APPLICATION "HCI.GRAPH_FACILITY -MEASUREMENT
  --            \MOTOR\CURRENT -GRAPH 1 -MINIMUM 0.0 -MAXIMUM 100.0", "",  false

  procedure START_UCL_DEBUGGER
    (in  ITEM   : UCL_AUTOMATED_PROCEDURE;
     in  NODE   : EGSE_NODE := \\;
     out STATUS : INTEGER);

  -- Description:
  --   Start UCL Debugger.
  -- Parameters:
  --   IN:
  --     ITEM   : UCL automated procedure loaded into debugger
  --     NODE   : Node the debugger will connect to
  --   OUT:
  --     STATUS : return status (POSIX error codes, 0 is success)

end HLCL_HCI_CMDS;
```

Copyright per DIN 34

J CGS SYSTEM LIMITATIONS

J-1 Diverse CGS Constraints

The following limitations exist for the size of system tables:

Number of enditems in ADU (Structured)	100 *)

NOTE: *) The number of enditems in a structured ADU is also limited by the size of the ADU (see below). In case of EGSE_BYTE_STREAM_MEASURMENTS (size = 256 characters effectively) not more than 15 enditems can be defined in a structured ADU otherwise the maximum size as defined below would be violated.
Furthermore the selection to have a physical address defined for each measurement reduces the maximum number of enditems to 68 only (between 68 and 100, dependent on the physicall address information given)

Number of online parameters for telecommands / binary packets	255
Number of predefined parameters for Predefined TC and Binary Packets	255
Number of online parameters for software commands	255
Size of a CCSDS Packet (bytes)	4096

Configurable in cgs_configuration.xml

Number of end items in a synoptic picture	50

parameter ONLINE_TEST_CONTROL.SYNOPTIC_DISPLAY.MAXVARIABLES

Size of the data part of an ADU (bytes)	4096
Number of statecodes per discrete calibration	256
Length of a calibrated String (bytes)	255
Number of nominal limit sets per end item	5
Number of entries in the system topology table	80
Number of end items in a monitor list	500
Number of end items in a GDU list	500
Number of entries in a user message description table (some 200 are used for CGS itself from a predefined file)	1500
Number of raw values in the simulated value table	1000
Number of ADUs of type UNSTRUCTURED or CCSDS in the simulated value table	100

Copyright per DIN 34

Number of CDUs loadable into each test node	500
Maximum Size of Source Code for Automated Procedures (bytes)	300 K
Maximum Size of I– Code at compile time (bytes)	(*) (>32) K

- *(*) More specific I–Code constaints are described in the I–Code reference manual.*

Maximum Size of I– Code at runtime for Automated Procedures (bytes)	400 K

- *Configurable in cgs_configuration.xml*

Max Number of Variables in Global Data Area of User Libraries and APs per compilation unit (depending on the variable types)	128..256

parameter TES.KERNEL.STACK_SIZE.UCL_INTERPRETER.

- *Configurable in cgs_configuration.xml*

Number of active APs on a test node	40

parameter TES.KERNEL.UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER.

Number of SAS on a test node	40

J-2 Miscellaneous Resources

During run time, Server disk space storage can be affected, eg by temporary files. Such temporary files will be created in specific file system temporary areas (eg /tmp). The temporary files will be created during standard process creation and during various software tasks eg file editing, compilation, temporary log files etc. The temporary areas should also allow for the dumping of system (UNIX) software during system errors. To cover for these eventualities, 100 MB to 200 MB should be allocated to the appropriate file systems to allow for temporary files in /tmp.

Copyright per DIN 34

K USER DEFINABLE CONFIGURATION FILES FOR CGS

K-1 Scope of Configuration Files

The CGS configurations files with name **cgs_configuration.xml** can have global, local or user scope. The location of the global configuration files are: $CGS_HOME/etc, while the location of the local configuration files are $CGS_HOME/local/config.

The user scope configuration files are located in $HOME/.cgs/config.

In a "client–server" CGS configuration all clients share the global and local file system. In a CGS "box configuration" a local file system can be locally defined. A box configuration shall have an own TRDB.

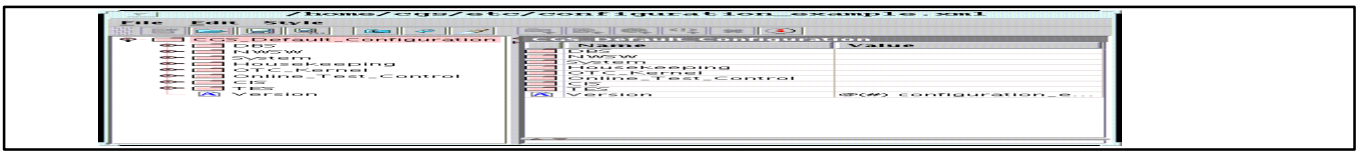
Global configuration files may be overwritten on local or user scope when stored in the identified locations.

K-2 Configuration Editor

For configuration purposes CGS provides a graphical editor in $CGS_HOME/gsaf/config/bin/common/configuration_editor

```
> $CGS_HOME/gsaf/config/bin/common/configuration_editor –help
--------------------------HELP---------------------------------
   Usage: configuration_editor [file] [–options]
   –admin –– allows to modify tree (default: change value only)
   –launch_bar file adds a user–defined launch bar
   –help ---- displays this output
---------------------------------------------------------------
```

$CGS_HOME/gsaf/config/bin/configuration_editor $CGS_HOME/etc/configuration_example.xml



Copyright per DIN 34

K-3 Config tool

For different work on configuratiions CGS provides a command line tool in $CGS_HOME/gsaf/cgsi/bin/linuxi with name config.

```
> $CGS_HOME/gsaf/cgsi/bin/linuxi/config -help ...

config                      <- configuration tool
  -help ["..."]             <- print help text ["..." = long]

  list                      <- list configuration
    <config_file>           <- pathname of config file
    -branch <value>...      <- branch(es) to be listed
    -comments               <- include comments

  export                    <- export configuration
    <config_file>           <- pathname of config file
    [<export_file>]         <- pathname of export file
    -branch <value>...      <- branch(es) to be listed

  import                    <- import configuration
    <config_file>           <- pathname of config file
    <export_file>           <- pathname of export file

  upgrade                   <- upgrade configuration

              set attribute values and hidden states in configuration 'new_config' as defined in
configuration 'config', save result as 'config'

    <config_file>           <- pathname of config file
    <new_config_file>       <- pathname of new config file
    -mapping <value>        <- pathname of mapping file
    -verbose                <- print all activities to stdout
    -all                    <- upgrade all attributes, including the hidden ones

              the default behaviour is to upgrade the attributes not hidden in config only (note that
the hidden state in new config has no effect)

    -forced                 <- always save the config, also in case of errors

              the default behaviour is to save the config only if no errors occured during upgrade

  transfer                  <- transfer configuration

              set attribute values and hidden states in configuration 'new_config' as defined in
configuration 'config', save result either as 'new_config' (the default), or, if given, as
'result_config'

    <config_file>           <- pathname of config file
    <new_config_file>       <- pathname of new config file
    [<result_config_file>] <- pathname of result config file
    -mapping <value>        <- pathname of mapping file
    -verbose                <- print all activities to stdout
    -all                    <- upgrade all attributes, including the hidden ones

              the default behaviour is to upgrade the attributes not hidden in config only (note that
the hidden state in new config has no effect)

    -forced                 <- always save the config, also in case of errors
```

Copyright per DIN 34

```
the default behaviour is to save the config only if no errors occured during upgrade
```

K-4 List of Configuration and Property Files

The following list shows the global CGS configuration and property files defined.

- CGS configuration file (configuration.xml)
  - contains entries of former otc_properties–orig.xml
  - contains entries of former TES_CONFIG_FILE (for parameter mapping use command $CGS_HOME/cgsi/bin/linuxi/tes–convert_tes_props_to_cgs –list)
  - contains entries of former cis_properties–orig.xml
  - contains entries of former tscv_configuration_file.dat.orig
  - contains entries of former tev_configuration_file.example
  - contains entries of former tev_configuration.data
  - contains entries of former tev data definition files
  - contains entries of former dbs_configuration_file.def
  - contains entries of former fa_sas_configuration_file.def

Configuration Files for TEV in Batch Mode

- data_list.def_syntax
- data_list_def.max_template_1
- data_list_def.max_template_2
- data_list_def.min_template
- data_set.def_syntax
- data_set_def.max_template_1
- data_set_def.max_template_2
- data_set_def.max_template_3
- data_set_def.max_template_4
- data_set_def.min_template
- event_list.def_syntax
- event_list_def.max_template
- event_list_def.min_template
- graph.def_syntax
- graph.min_template
- one_shot_graph.def_syntax
- one_shot_graph.min_template

Copyright per DIN 34

- raw_data_dump.def_syntax
- raw_data_dump_def.max_template_1
- raw_data_dump_def.max_template_2
- raw_data_dump_def.min_template
- statistics.def_syntax

Copyright per DIN 34

K-5 Example Configuration and Property Files

• CGS configuration file (cgs_configuration_defaults.xml)

```
> $CGS_HOME/gsaf/cgsi/bin/linuxi/config list $CGS_HOME/etc/cgs_configuration_defaults.xml –comments
```

which following symbols/keywords:

+	group
–	attribute
[class]	class [Administrator|User]
<visibility>	normaly all attributes are hidden and the software components use the defaults as listed below. To change an attritute, unhide it in configuration_editor and change the value.
= default	default value

```
+ Nwsw [Administrator]

  This group contains configuration parameters of the NWSW.
  Do not alter them without contacting CGS engineering.

  – Receive_Task_Stack_Size [Administrator] = 200000 <hidden>

    NWSW PEER–TO–PEER CONNECTION SERVICES
    Stacksize in bytes of the receive tasks in the NWSW.
    Lower limit: 100000.

  – Accept_Task_Stack_Size [Administrator] = 200000 <hidden>

    NWSW PEER–TO–PEER CONNECTION SERVICES
    Stacksize in bytes of that NWSW task, that handles the incoming connection requests.
    Lower limit: 100000.

  – Udp_Time_Betw_Alive_Msgs [Administrator] = 3 <hidden>

    NWSW EVENT–DISTRIBUTION SERVICES
    Defines the maximum time of inactivity in seconds for an event generator.
    If no messages are sent for this timespan, the underlying NWSW will generate
    an ALIVE message to indicate the healthy state of this sender
    Lower limit: 1.

  – Size_Of_Bc_Tcp_Sock_Bufs [Administrator] = 307_200 <hidden>

    NWSW EVENT–DISTRIBUTION SERVICES
    Defines the size in Bytes of the 2 socket buffers for the conection from
    the event–distributor–daemon to the client. The higher this value is, the
    more messages are buffered to survive data peaks.
    THE DAEMON CLOSES THE CONNECTION, IF THE CLIENT IS NOT READING ITS MESSAGES
    IN TIME !!!!!
    Lower limit: 65536

  – Daemons_Udp_Server_Port [Administrator] = 11001 <hidden>

    NWSW EVENT–DISTRIBUTION SERVICES
    Defines the port on which the event–distributor daemon is waiting for
    events or messages coming via UDP (in Broadcast and in Multicast mode)

  – Daemons_Tcp_Server_Port [Administrator] = 11001 <hidden>

    NWSW EVENT–DISTRIBUTION SERVICES
    Defines the port on which the event–distributor daemon is waiting for
    connection requests from clients (in Broadcast and in Multicast mode)

  – Daemon_Use_Multicast [Administrator] = True <hidden>

    NWSW EVENT–DISTRIBUTION SERVICES
    Whether Multicast should be used to distribute events and messages.
    If set to False, simple UDP Broadcast is used.
    Allowed Values: True|False
```

Copyright per DIN 34

```
- Daemon_Use_Multicast_Group [Administrator] = 228.0.0.10 <hidden>

  NWSW EVENT-DISTRIBUTION SERVICES
  Only used, if Nwsw.Daemon_Use_Multicast is set to True !!
  Defines the multicast 'channel', which is used to distribute events and
  messages.
  Allowed Values: 225.0.0.0 .. 238.255.255.255

- Daemon_Multicast_TTL_Value [Administrator] = 5 <hidden>

  NWSW EVENT-DISTRIBUTION SERVICES
  Only used, if Nwsw.Daemon_Use_Multicast is set to True !!
  Defines how long resp. how far a multicast packet is distributed
  Allowed Values: 0 .. 255

- Ignore_Alive_Msg_Check [Administrator] = false <hidden>

  NWSW IGNORE-ALIVE-MSG-CHECK
  This switch is for debugging purposes.
  If it is true, the Cb_Alive_Msg_To_Occ will not be called.
  A message will be written to stdout instead.

      Range:                true/false
      Recommended value:    false

- Max_Transmission_Delay [Administrator] = 30 <hidden>

  NWSW MAX-TRANSMISSION-DELAY
  Maximum transmission/detection delay in seconds.
  Maximum duration between two messages, before an alive-msg-timeouts
  occurs.

      Range:                Positive [s]
      Recommended value:    30

- Default_Transmission_Delay [Administrator] = 10 <hidden>

  NWSW DEFAULT-TRANSMISSION-DELAY
  Default transmission/detection delay in seconds.
  Duration after that an alive-msg-timeouts callback is activated.
  The sum of Nwsw.Max_Transmission_Delay and this value define the timeout
  used to indicate receiver alive-msg-timeouts.

      Range:                1 .. Nwsw.Max_Transmission_Delay [s]
      Recommended value:    10

- Connect_Timeout [Administrator] = 1000 <hidden>

  NWSW CONNECT-TIMEOUT
  Default connect timeout in milli seconds.
  Maximum time that a client waits for the availibility of a sleeping server.
  This timeout is for the case, that a server is up but not longer reacting
  on connection requests.
  If a server is not up at all, a connect call returns immediately waithout waiting
      Range:                Positive [ms]
      Recommended value:    1000

- Alv_Msg_Poll_Interval [Administrator] = 2000 <hidden>

  NWSW ALIVE-MSG-CHECK SERVICES
  Defines the interval between two checks for Alive-Msg-Timeouts in [ms]
  Lower limit: 100

- Proc_Alv_Msg_To_Task_Stack_Size [Administrator] = 200000 <hidden>

  NWSW ALIVE-MSG-CHECK SERVICES
  Stacksize in bytes of that NWSW task, that executes the Alive-Msg-Notification callbacks.
  Lower limit: 100000.

- Default_Socket_Buffer_Size [Administrator] = 40 <hidden>

  NWSW Default_Socket_Buffer_Size
```

Copyright per DIN 34

```
    Default sizes for the sockets send and receive buffer in kilo Byte.

        Range:                Positive
        Recommended value:    40

+ Resource

  + Tasking

    - UI_Task_Stack_Size = 2_000_000 <hidden>

+ System

  - MaxTestNodes [Administrator] = 32 <hidden>

    Maximum number of Test Nodes.
     !!! Caution: it also means that a test node index cannot be greater than the value of MaxTestNodes.
For example: MaxTestNodes is 10, so TES_11 is not a valid Test Node!

  - MaxWorkstations [Administrator] = 32 <hidden>

    Maximum number of Workstations (HCI and CIS)

  + Time

    + TAI_Epoch

      - Start_Year [User] = 1_980 <hidden>

        (U) The CCSDS-Recommended epoch start time (TAI)

            Range:                integer'range
            Recommended value:    1_980

      - Start_Month [User] = 1 <hidden>

        (U) The CCSDS-Recommended epoch start time (TAI)

            Range:                integer'range
            Recommended value:    1

      - Start_Day [User] = 6 <hidden>

        (U) The CCSDS-Recommended epoch start time (TAI)

            Range:                integer'range
            Recommended value:    6

  + Forced_Exit_Status [Administrator]

    This group allows to force a specific exit status to be returned from one or more processes.
      The exit status given in attribute 'Status' is imposed on the processes given in attribute
'Processes'.
    If empty, standard exit status handling is used.

    - Status [Administrator] =  <hidden>

      The forced exit status to be imposed on the processes given in attribute 'Processes'
      (a number in the range 0 .. 255).
      If empty, standard exit status handling is used.

    - Processes [Administrator] = () <hidden>

      The forced exit status given in attribute 'Status' is imposed on the processes listed here.
      Processes must be given with their process file name (base name),
      '*' selects all processes.

+ Data_System

  + CCSDS

    - Extended_ApId [User] = true <hidden>
```

Copyright per DIN 34

```
      (U) indicates if the packet type field in COF specific CCSDS primary header
            a) should be considered for calculating of sequence count
               (internal operation)
            b) should be set during setting APID
               (via system library command SET_CCSDS_APID)
               This enlarge the APID range from 0 .. 2^11 – 1 to
                                                0 .. 2^12 – 1.
           or not

          Range:                true/false
          Recommended value:    true

+ Housekeeping [User]

(U) User defined housekeeping values (HK).
    This housekeeping values can be set only by special application
    software (SAS) via TES_API call Set_Hk_Value
  or
    they are static string values from the CGS configuration (type = CONFIG).

    The behaviour is like normal housekeeping values.
    For each name (element) of the Names following attributes
        housekeeping Id    (integer in range 1130 .. 1200),
        housekeeping type  (for the first only STATE_CODE and
                                               CONFIG are allowed) and
        housekeeping value (initial value –
                            for STATE_CODE a string <= 8 characters,
                            for CONFIG     a string denotes a valid TES configuration parameter,
                                           which value shall be taken)
        and optionally a state code list
    needs to be specified.

    User defined HK values can be visualize in HCI window footers.
    For detailed information see description of Housekeeping.StatusDataFormat.

    example:

      Housekeeping.Names shall contain two elements USER_DEF_HK_1 USER_DEF_HK_2,
      which shall be displayed on HCI window footer like
      Downlink: <USER_DEF_HK_1>, Check Checksum: <USER_DEF_HK_2>

        In configuration editor: select Housekeeping
                                 select StatusDataFormat
                                 select Value field and enter: Downlink: %1130%, Check Checksum: %1200%
                                 select Names
                                 select add values (V+)
                                 select Value field and enter: USER_DEF_HK_1 (first element of example)
                                 select add values (V+)
                                 select Value field and enter: USER_DEF_HK_2 (second element of example)

      USER_DEF_HK_1 is housekeeping Id    = 1130,
                       housekeeping type  = STATE_CODE,
                       housekeeping value = INITIAL1,
        In configuration editor: select Housekeeping
                                 select add attribute (A+)
        In attribute = value   : select list of values
                                 select Name field and enter: USER_DEF_HK_1
                                 select Value field and enter: 1130
                                 select add values (V+)
                                 select Value field and enter: STATE_CODE
                                 select add values (V+)
                                 select Value field and enter: INITIAL1
                                 select Add attribute
                                 select add values (V+)
                                 select Value field and enter: INITIAL1, OK, ERROR
                                 select Add attribute

      USER_DEF_HK_2 is housekeeping Id    = 1200,
                       housekeeping type  = CONFIG,
                       housekeeping value = TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM,
        In configuration editor: select Housekeeping
```

Copyright per DIN 34

```
                                select add attribute (A+)
        In attribute = value    : select list of values
                                  select Name field and enter: USER_DEF_HK_2
                                  select Value field and enter: 1200
                                  select add values (V+)
                                  select Value field and enter: CONFIG
                                  select add values (V+)
                                                                    select  Value  field  and  enter:
TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM
                                  select Add attribute
        USER_DEF_HK_2 contains the static value from CGS configuration parameter as string.
```

- StatusDataFormat [User] = <hidden>

  (U) Visualization in HCI window footers
      HK values from MTP TES can be displayed in HCI window footers. To do this, specify
      Housekeeping.StatusDataFormat. The format is:
          {text}[%<hk id>%{text}]+
      where text is a free text and <hk id> is a valid housekeeping identifier.

      example:

        To display HK ID 3 (TES_MODE) and user defined HK ID 1130 (should be defined),
        from MTP specify in Housekeeping.StatusDataFormat:
           The TES mode is %3% and my HK 1130 is %1130%
        %3% and %1130% will be replaced by the actual values.

- Names [User] = () <hidden>

  (U) User defined housekeeping values names.
      Housekeeping.Names contains the names of user defined housekeeping values.

      example:

        Define two elements USER_DEF_HK_1 USER_DEF_HK_2 as follows
          In configuration editor: select Housekeeping
                                   select Names
                                   select add values (V+)
                                         select Value field and enter: USER_DEF_HK_1 (first element of
example)
                                   select add values (V+)
                                        select Value field and enter: USER_DEF_HK_2 (second element of
example)

      Do not forget to define for each name (element) of the Names following attributes:
          housekeeping Id   (integer in range 1130 .. 1200),
          housekeeping type (for the first only STATE_CODE and
                                                 CONFIG are allowed) and
          housekeeping value (initial value –
                              for STATE_CODE a string <= 8 characters,
                              for CONFIG   a string denotes a valid TES configuration parameter,
                                             which value shall be taken)

To apply new housekeeping variables in CGS Tools and USS Editor, start “CGS – Administration – Configuration – HK XML SCOE Generator” from CGS Start Center.

+ CLS

  Configuration group for the CGS Language System.

  + Dependencies

    This group contains options related to CLS dependency handling.

    - Load_Tree [Administrator] = true <hidden>

      Load complete dependency tree from database in one step.
      If false, load items in signle steps.

    - Load_Exclude [Administrator] = true <hidden>

      Exclude items already loaded from tree load step.
      If false, load complete dependency tree.

Copyright per DIN 34

```
  + Debug

    This group contains logging options for debugging purposes.

    – Log_Configuration [User] = false <hidden>

      Output the CLS configuration tree at program start.

    – Log_Dependencies [User] = false <hidden>

      Log CLS dependency checking events.

    – Log_Editor_Control [User] = false <hidden>

      Log various CLS Editor control events.

    – Log_Allocation [User] = false <hidden>

      Log allocation/deallocation events in a file <program_name>.alloc.

    – Log_Multiple_Deallocation [User] = false <hidden>

      Log multiple deallocations in a file <program_name>.alloc.

    – Deallocation_Alarm [User] = false <hidden>

      Print alarm messages for multiple deallocations.

+ DBS

  + RPI

    This group contains the configuration parameters specific to DBS.

    – DBS_PRINTER_1 = lp <hidden>

      Declaration of first printer to be used by DBS.

    – DBS_PRINTER_2 = lp <hidden>

      Declaration of second printer to be used by DBS.

    – SEND_MESSAGE_AGAIN_TIME = 300 <hidden>

      This value is used to avoid cyclic sending of messages to the
      message_handler. Same messages from the same destination will be Dropped,
      if they have already been sent 3 times within the last SEND_MESSAGE_AGAIN_TIME
      seconds. The value is expressed in seconds, i.e. a value of 300 means 5 minutes.

          Recommended value:     300

    – TN_EVT_BUFFER_SIZE = 2000 <hidden>

      Number of buffered events, used for testnodes. Only one buffer is maintained.

          Recommended value:     2000

    – MAX_EVT_IN_ERROR_CASE = 50000 <hidden>

      In case a communication error occurs during sending of the current EVT-file
      from local to central DBS the file will be re-used, i.e. EVT-values produced
      afterwards will be appended to this file in order to minimize the number of
      EVT-files on the testnode. The following value denotes the maximum number of
      entries within the local EVT-file before it is closed. If it can never be sent
      to central DBS a warning message will be produced to inform the user to handle
      this file via the DBS recovery scripts.

          Recommended value:     50000

    – MAX_EVT_NUMBER_IN_LOCAL_FILE = 1000 <hidden>

      Max Number of Events in a local file (before it is sent to central).
```

Copyright per DIN 34

```
        Recommended value:     1000

    - ONL_EVAL_EVT_PERIOD = 10 <hidden>

      The period (in seconds) between local EVT file storages on the Central
      DBS server. Then they will be available for online evaluation.

        Recommended value:     10

    - EVL_BUFFER_SIZE = 2000 <hidden>

      Number of buffered Engineering Value Logs, used for testnodes. Only one buffer is maintained.

        Recommended value:     2000

    - MAX_EVL_NUMBER_IN_LOCAL_FILE = 2000 <hidden>

      Maximum Number of Engineering Values in EVL Local Files.

        Recommended value:     2000

    - MAX_EVL_IN_ERROR_CASE = 50000 <hidden>

      In case a communication error occurs during sending of the current EVL-file
      from local to central DBS the file will be re-used, i.e. EVL-values produced
      afterwards will be appended to this file in order to minimize the number of
      EVL-files on the testnode. The following value denotes the maximum number of
      entries within the local EVL-file before it is closed. If it can never be sent
      to central DBS a warning message will be produced to inform the user to handle
      this file via the DBS recovery scripts.

        Recommended value:     50000

    - ONL_EVAL_EVL_PERIOD = 10 <hidden>

      The period (in seconds) between local EVL file storages on the Central
      DBS server. Then they will be available for online evaluation.

        Recommended value:     10

    - PERCENT_OF_FREE_SPACE = 2 <hidden>

      This parameter is used to define the TRDB status DISK_FULL. In case the free disk space
      of the disk partition defined by $VICOS_CEN_DBS_HOME is less than PERCENT_OF_FREE_SPACE then
      the TRDB disk status is set to DISK_FULL. An appropriate message will be displayed in the
      message handler.

        Recommended value:     2

    - AUTO_ARCHIVE_EVENT_TABLE_SIZE = 10000000 <hidden>

       In case of auto-archived sessions the Event Table will be flushed into a file, if it exceeds this
limit.
      This file is then requested to be stored inside TRDB (part of auto archiving).
      The value is expressed in bytes.

        Recommended value:     10000000

    - ORA_EVENT_BUFFER_SIZE = 300 <hidden>

      Number of Events in an Oracle buffer to accelerate the event deletion and insertion
      (the events are deleted or inserted using the ORACLE array-fetch mechanism, i.e. an array of events
      is passed to ORACLE rather than a single event).
      Warning: this value directly affects the stack size of the DBS processes!

        Recommended value:     300

    - MAX_EVL_FILE_SIZE = 1000000 <hidden>

      Maximum size of a Central EVL file stored into the TRDB (expressed in bytes).
      Another EVL file will be created, if the size of the current file reaches this value.
```

Copyright per DIN 34

```
        Recommended value:     1000000

  - CRITICAL_SPACE = 4 <hidden>

    The parameters CRITICAL_SPACE and SECURE_SPACE are used to define the level at which
    automatic archiving will start and stop.  If CRITICAL_SPACE = 4 and SECURE_SPACE = 6
    then automatic archiving will start when free disk space is 4% of the total disk space
    and automatic archiving will stop when free disk space is 6% of the total disk space
    Note SECURE_SPACE must always be greater than CRITICAL_SPACE.

        Recommended value:     4

  - SECURE_SPACE = 6 <hidden>

    The parameters CRITICAL_SPACE and SECURE_SPACE are used to define the level at which
    automatic archiving will start and stop.  If CRITICAL_SPACE = 4 and SECURE_SPACE = 6
    then automatic archiving will start when free disk space is 4% of the total disk space
    and automatic archiving will stop when free disk space is 6% of the total disk space
    Note SECURE_SPACE must always be greater than CRITICAL_SPACE.

        Recommended value:     6

  - NUMBER_OF_FA_DRIVE_AVAILABLE = 2 <hidden>

    This value denotes the number of final archive partitions. The FA SAS (Final Archive
    Special Application Software) is used to archive, retrieve, export and import sessions.
    The partitions to be used are declared by values FA_DEVICE_FILENAME1 and FA_DEVICE_FILENAME2.
    Only a maximum of 2 partitions is maintained.

        Recommended value:     2

+ Command_History

  This group contains the configuration parameters specific to the command history.

  - DBS_COMMANDS_PER_REPORT_SLICE = 20 <hidden>

    Number of entries contained within one XML report file. The files are generated
    by the command history and sent via CIS to the MCS Tools on request.

        Recommended value:     20

  - DBS_UDP_BROADCAST_ENABLED = true <hidden>

    Enables subscription messages from the command history to be sent to
    the CIS clients via UDP broadcast resp. multicast. Value False will force the
    command history to send the messages one by one via TCP/IP to each connected client.
    Note: message distribution method UDP broadcast or multicast may be switched using
    the configuration parameter Nwsw.Daemon_Use_Multicast. See description of that
    parameter for details.

        Range:                 true/false
        Recommended value:     true

+ FA_SAS

  This group contains the configuration parameters specific to the DBS
  Final_Archive Special Application Software (FA_SAS)

  - FA_DEVICE_FILENAME1 = /dev/mo1 <hidden>

    Location within the filesystem for the FA_SAS files, i.e. exported and/or archived sessions
    The specified partition must be visible on the DB server.
    Only a maximum of 2 locations/partitions is maintained.

  - FA_DEVICE_FILENAME2 = /dev/mo2 <hidden>

    Location within the filesystem for the FA_SAS files, i.e. exported and/or archived sessions
    The specified partition must be visible on the DB server.
    Only a maximum of 2 locations/partitions is maintained.
```

Copyright per DIN 34

```
    - DEFAULT_DEVICE_FOR_AUTOMATIC_ARCHIVING = 1 <hidden>

      This value denotes the device which will be selected for automatic archiving in batchmode,
      i.e. without user interrogation.

          Valid range:           1..2
          Recommended value:     1

    - FA_SAS_PRINTER = lp <hidden>

      The Name of the environment variable defining the FA SAS Printer
      (used by MMI operation Print Detailed List).

    - PERCENT_OF_FREE_SPACE = 1 <hidden>

      Minimum percentage of free space in the FA partition needed to work.

          Recommended value:     1

+ OTC_Kernel

  + Communication

    - DataSendRetries = 2 <hidden>

      Count of retries of data sending in case of send problems for an existing connection.
      DataSendRetries = 0 is no retries, the data send only once.

          Range:                 natural'range
          Recommended value:     2

    - DataSendRetryDelay = 0.5 <hidden>

      Delay in seconds between different retries (defined by configuration parameter DataSendRetries)
      of data sending in case of send problems for an existing connection.

          Range:                 duration'range
          Recommended value:     0.5 [s]

    - DataRequestRetryDelay = 60.0 <hidden>

       If data can't be requested (e.g. for the AP Status) Online Test Control will retry the request
after a.m. delay.

    - DataRegistrationTimeOut = 60 <hidden>

       Defines the time in seconds HCI will wait for the test node until it registers its data (data
distribution table).

    - EndItemTimeOut = 20.0 <hidden>

      Minimum number of seconds Online Test Control will wait until an end item value is delivered from a
test node (HLCL commanding).

    - APClientTimeOut = 2.0 <hidden>

    - LoadSynopticTimeOut = 20_000 <hidden>

      Defines the time in milliseconds TES will wait until the display_picture library call is timed out.

    - RemoveSynopticTimeOut = 5_000 <hidden>

      Defines the time in milliseconds TES will wait until the remove_picture library call is timed out.

    - USSResponseTimeOut = 25_000 <hidden>

      Defines the time in milliseconds Online Control will wait for USS responses.

    - StartProgramTimeOut = 20_000 <hidden>

      Defines the time in milliseconds TES will wait until the start program call is timed out
```

Copyright per DIN 34

```
    – DataTimeOut = 0.4 <hidden>

      Time out of a data message in seconds.

    – ResponseTimeOut = 0.8 <hidden>

      Time out of a response message in seconds.

    – SocketBufferSize = 102400 <hidden>

      Sets the socket buffer size use for CGS–CGS communication.

  + Resources

    – MaxAPCommands = 10 <hidden>

       Maximum number of AP commands (e.g. start/abort AP, execute library routine) that can be executed
in parallel.

    – MaxDeliveryNotes = 500 <hidden>

      Maximum number of delivery notes (data requests and cancel notes) that can be handled.

    – MaxDeliveryNotesItems = 5000 <hidden>

    – MaxEndItemRequests = 10 <hidden>

      Maximum number of enditems that can be queried by HLCL commands at the same time.

    – MaxDataPoolDeliveries = 200 <hidden>

      Maximum number of deliveries to store in the delivery pool for freeze mode.

    – EnditemListSize = 4_000 <hidden>

               Defines the number of enditems that can be stored in a message of type
ENDITEMS_MESSAGE/APPEND_ENDITEMS (data distribution table/ announce enditems).

    – SMT_Refresh = 0.333 <hidden>

      Refresh rate in seconds of SMT reading.

    + Tasking

      – Data_Distributor_Interim_Buffer_Task_Size = 48_000 <hidden>

      – T_OBCS_Data_Distributor = 200_000 <hidden>

      – Command_Distributor_T_Server = 200_000 <hidden>

      – T_HLCL_Interpreter = 300_000 <hidden>

      – T_HLCL_Interposer = 300_000 <hidden>

+ Online_Test_Control

  + HLCL

    – LoginSequence = $HOME/.cgs/hlcl_login.seq <hidden>

      Location of the HLCL login sequence

  + Log

    – LoadErrorMessages = True <hidden>

       Controls loading of error message definitions. If disabled error messages will neither be loaded
from file nor from data base.

    – LoadErrorMessagesFromDB = False <hidden>

        Controls loading error message definitions from data base. If disabled error messages are only
```

Copyright per DIN 34

```
loaded from file.

    - CodeOffset = 5000 <hidden>

      Offset added to the internal error codes of Online Test Control.

    - MessageFile = ${CGS_HOME}/etc/messages.def <hidden>

      Defines the location of the message definition file.

    - UserName = * <hidden>

       Defines the user name of the addressed message window, default is all users (”” is same user as
Online Test Control, ”*” is all users).

  + Detailed_Info

    - WindowMemoryMaximum = 5_000 <hidden>

      Memory used by detailed info window.

    - Padding_Spaces = 3 <hidden>

      Padding spaces between key and value in the detailed info window.

    - Not_Available_Text = <Not available> <hidden>

      Text for values which are not available.

  + General

    - ExitTimeOut = 45 <hidden>

      Time out after which Online Test Control will ask for a forced shut down.

    - LogoFile = ${CGS_HOME}/etc/logo.${CGS_ARCH} <hidden>

      Logo file name.

    - MaxApplicationWindows = 50 <hidden>

      Maximum number of window applications that can be executed in parallel.

  + Resources

    - Max_OTC_Applications = 1_000 <hidden>

      Maximum Online Test Control applications that can be executed in parallel.

    - SynopticToolkit = GIPSY <hidden>

      Selection of toolkit used by synoptics, values are GIPSY or DATAVIEWS; default is GIPSY

    + Tasking

      - Cmd_Window_Manager = 300_000 <hidden>

      - Explorer_T_Create_Task = 400_000 <hidden>

      - ScreenSetupLoader = 200_000 <hidden>

      - T_ANSWER_TASK = 50_000 <hidden>

      - Input_Dialog_T_Input_Sender = 128_000 <hidden>

      - Input_Dialog_T_Input_OBCS = 128_000 <hidden>

  + Windows

    - PathnameStoredLength = 200 <hidden>

        Defines the maximum number of character for pathname in input text fields (e.g. name of the
```

Copyright per DIN 34

measurement in the Graph Facility Properties dialog)

- PathnameDisplayLength = 200 <hidden>

  Defines the maximum number of character for pathname displayed in input text fields.
  If the pathname exceeds the display length scroll–buttons are added to the text field.

+ Color

  Decription of Color Specifications
  Colors can be specified by using color name defined in the color database (use Unix showrgb) to look up color database) or hexadecimal code.

  The string can be one of :
  – ”RBG:FF/FF/FF” where the ”FF” substrings are respectively the value of the red, green and blue components.
  – ”color_name” which can be any color name defined in the file rgb.txt of the user’s system. You should always check that Wrong_Color was not raised, in case the color was not known on the user’s system. This string is case insensitive. Color names are not supported on Windows systems.

- MonitoringDISABLED = turquoise <hidden>

- MonitoringIN_LIMITS = green <hidden>

- MonitoringNOMINAL_LIMIT_VIOLATION = yellow <hidden>

- MonitoringNOMINAL_LOW_LIMIT_VIOLATION = yellow <hidden>

- MonitoringNOMINAL_HIGH_LIMIT_VIOLATION = yellow <hidden>

- MonitoringDANGER_LOW_LIMIT_VIOLATION = red <hidden>

- MonitoringDANGER_HIGH_LIMIT_VIOLATION = red <hidden>

- MonitoringUNDEFINED = rgb:c7/89/00 <hidden>

- AcquisitionStatus_REQUESTED = purple <hidden>

- AcquisitionStatus_NOT_MAINTAINED = purple <hidden>

- AcquisitionStatus_NOT_ACQUIRED = purple <hidden>

- AcquisitionStatus_NOT_RECEIVED = purple <hidden>

- AcquisitionStatus_INVALID = purple <hidden>

- AcquisitionStatus_ACQUIRED = white <hidden>

- AcquisitionStatus_DATA_INTERRUPTION = purple <hidden>

- AcquisitionStatus_STATIC = cyan <hidden>

- ProcessingDisabled = cyan <hidden>

- Highlight_Fg_Color = white <hidden>

  Defines the foreground color for data text highlighting of the raw data dump window.

- Highlight_Bg_Color = blue <hidden>

  Defines the background color for data text highlighting of the raw data dump window.

+ AP_Status

- MaxWindows = 2 <hidden>

  Defines the maximum number of AP Status Displays that can be displayed in parallel.

- UpdateRate = 5 <hidden>

  Defines how often the AP status window is updated (in seconds); the value 0 means it will be updated only if one or more of the values are changed.

Copyright per DIN 34

```
  + CGS_Tools

    - StartScript = ${CGS_HOME}/gsaf/hci/bin/common/jotc.sh <hidden>

      Location of CGS Tools start script.

  + Clock

    - ReplayFooter = Yes <hidden>

      Controls display of window footer when started in replay mode.

    - UpdateRate = 5 <hidden>

      Defines how often the Clock window (replay mode only) is updated (in seconds); the value 0 means it
will be updated only if one or more of the values are changed.

    - LocalTimeTCI = LT <hidden>

      Time Code Identifier for local time.

    - SMT_TCI = SMT <hidden>

      Time Code Identifier for simulated mission time.

  + Command_Facility

    - History = 200 <hidden>

      Defines the size of the history buffer, i.e. the number of commands stored in the history.

    - DontLog = ? LIST <hidden>

      Specifies a list of commands not to be logged (this is applicable for HLCL command windows, but not
synoptic displays).
      The list may include:
      – a command/procedure name for primary commands,
      – a reserved word (import, type) for commands that start with a reserved word,
      – a pathname for APs and command sequences from MDB,
      – a qualified name (pathname.identifier) for UCL library procedures,
      – a file name in string quotes for command sequences from files,
      – "?" for the ? command,
      – ":=" for assignments
      – "*" for all commands (to disable logging)
      LOG_SYNTAX_ERRORS en/disables logging for syntactically wrong commands.
      If logging is disabled with DONT_LOG = "*" nothing will be logged even if LOG_SYNTAX_ERRORS is set.

    - CommandLength = 450 <hidden>

    - Prompt = HLCL:  <hidden>

  + Detailed_Info_Window

    - WindowMemoryMaximum = 100_000 <hidden>

      Memory used by detailed info window.

  + Explorer

    - ButtonWidth = 128 <hidden>

      Pixel width of the buttons.

    - ButtonXOffset = 64 <hidden>

      Pixel offset for a new tree level related to its parent level.

    - ButtonYOffset = 1 <hidden>

      Pixel distance of buttons in y direction.
```

Copyright per DIN 34

+ Out_Of_Limit_Display

  – MaxWindows = 8 <hidden>

  – DefaultDirectory = ${HOME}/.cgs/screen_setup_pool <hidden>

Default directory where the file chooser of the Out_Of_Limit display will search for its configurations.

+ Graph_Facility

  – MeasurementDisplayLength = 60 <hidden>

  Display length of the measurement input field.

  – MaxWindows = 4 <hidden>

  Defines the maximum number of Graph Facilities that can be displayed in Parallel.

  + Templates

  – Names = (BarChart, LineGraph, LineValueGraph, StripChart, Text) <hidden>

Names enumerates the templates used for Graph Facility. Each name must define an attribute with three values: label, template filename, graph has history. Label is the label of the graph selector button. Template filename describes the GWDU generated template (see User Manual how to create a template). Graph has history indicates whether the graph is able to display more than one sample.

  – BarChart = (Bar Chart, bar_chart.dv, TRUE) <hidden>

  – LineGraph = (Line Graph, line_graph.dv, TRUE) <hidden>

  – LineValueGraph = (Line/Value Graph, line_value_graph.dv, TRUE) <hidden>

  – StripChart = (Strip Chart, strip_chart.dv, TRUE) <hidden>

  – Text = (Text, text.dv, FALSE) <hidden>

  – DefaultSlots = 10 <hidden>

  Number of slots used as default in graph facility.

  – TemplatesDirectory = ${CGS_HOME}/etc/templates/gf_black_background <hidden>

  Directory where to find the Graph Facility templates.

  – DefaultColors = 0, 204, 255; 0, 0, 255; 155, 0, 255; 255, 0, 255; 255, 0, 192; 255, 0, 132; 255, 0, 0; 255, 98, 0 <hidden>

Comma/semicolon separated list of RGB values for each measurement that can be loaded into Graph Facility (in decimal)

  – LineTypeDirectory = ${CGS_HOME}/etc/bitmaps/line_types <hidden>

  Directory where to find the line type bitmaps.

+ Input_Dialog

  – MaxWindows = 10 <hidden>

  Maximum number of AP Input Dialog windows.

  – InputFieldDisplayLength = 20 <hidden>

  Display length of the input field.

+ Main_Menu

  – EnableDisconnectedNodes = Off <hidden>

  If on disconnected nodes are selectable on the execution node submenu.

Copyright per DIN 34

+ Raw_Data_Dump

  – Program = ${HCI_HOME}/bin/${CGS_ARCH}/otc_rdd <hidden>

  – MaxBriefmodePackets = 200 <hidden>

    Defines the maximum number of displayed brief mode packets.

  – DefaultDirectory = ${HOME} <hidden>

    Default directory where the file chooser of the raw data dump window will saved the current raw data dump packet.

  – PacketDefault = <hidden>

    Default packet pathname, displayed when the properties are poped up.

  + PropertiesDialog

    – BaseDefault = 16 <hidden>

      Defines the base–button selected by default.

    – BaseCount = 5 <hidden>

      Defines the number of base–buttons.

    – Base1 = 2 <hidden>

      BaseN define the base choosen after selection of base–button N, e.g. after selection of base–button 2 the output is displayed in octal format. A base between 128 and 255 is interpreted as ASCII.

    – Base2 = 8 <hidden>

      BaseN define the base choosen after selection of base–button N, e.g. after selection of base–button 2 the output is displayed in octal format. A base between 128 and 255 is interpreted as ASCII.

    – Base3 = 10 <hidden>

      BaseN define the base choosen after selection of base–button N, e.g. after selection of base–button 2 the output is displayed in octal format. A base between 128 and 255 is interpreted as ASCII.

    – Base4 = 16 <hidden>

      BaseN define the base choosen after selection of base–button N, e.g. after selection of base–button 2 the output is displayed in octal format. A base between 128 and 255 is interpreted as ASCII.

    – Base5 = 128 <hidden>

      BaseN define the base choosen after selection of base–button N, e.g. after selection of base–button 2 the output is displayed in octal format. A base between 128 and 255 is interpreted as ASCII.

    – BytesPerLineDefault = 32 <hidden>

      Bytes–per–line–button selected by default.

    – BytesPerLineCount = 8 <hidden>

      Number of bytes–per–line–buttons selected by default.

    – BytesPerLine1 = 8 <hidden>

      BytesPerLineN define the bytes per line choosen after selection of bytes–per–line–button N

    – BytesPerLine2 = 16 <hidden>

      BytesPerLineN define the bytes per line choosen after selection of bytes–per–line–button N

Copyright per DIN 34

- BytesPerLine3 = 24 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

- BytesPerLine4 = 32 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

- BytesPerLine5 = 40 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

- BytesPerLine6 = 48 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

- BytesPerLine7 = 56 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

- BytesPerLine8 = 64 <hidden>

  BytesPerLineN define the bytes per line choosen after selection of bytes-per-line-button N

+ SAS_Status

- MaxWindows = 4 <hidden>

  Defines the maximum number of SAS Status Displays that can be displayed in parallel

- UpdateRate = 0 <hidden>

  Defines how often the SAS status window is updated (in seconds); the value 0 means it will be updated only if one or more of the values are changed.

+ Synoptic_Display

- MinimumWindowSize = 50 <hidden>

  Minimum window size accepted for synoptic displays.

- MaxVariables = 500 <hidden>

  Maximum number of variables that can be displayed in one synoptic.

- MaxStatusTextLength = 40 <hidden>

  Maximum length of the status text info field.

- DisplayNotMaintainedItems = Disable <hidden>

  If enabled all items of a synoptic display that are not maintained by any execution node are reported to the message handler

- NACQFlagBackgroundRed = 0 <hidden>

  Define the background color of the ’data acquisition’ flag as RGB value.Value for red.

- NACQFlagBackgroundGreen = 0 <hidden>

  Define the background color of the ’data acquisition’ flag as RGB value.Value for green.

- NACQFlagBackgroundBlue = 0 <hidden>

  Define the background color of the ’data acquisition’ flag as RGB value.Value for blue.

- NACQFlagForegroundRed = 255 <hidden>

  Define the foreground color of the ’data acquisition’ flag as RGB value.Value for red.

- NACQFlagForegroundGreen = 255 <hidden>

Copyright per DIN 34

Define the foreground color of the ’data acquisition’ flag as RGB value.Value for green.

– NACQFlagForegroundBlue = 0 <hidden>

Define the foreground color of the ’data acquisition’ flag as RGB value.Value for blue.

– NACQFlagTextSize = 2 <hidden>

Defines the text size of the ’data not acquired’ flag, 1 is very small, 9 very large.

+ Flag_Text

Status flags for synoptic display output objects.
Each status flag text has a priority in case more than one measurement is attached to an output object (e.g. a bar chart displaying \MEA_1 and \MEA_2).
Synoptic Displays uses a priority list to determine the flag text, 0 is highest priority, 7 lowest (i.e. if \MEA_1 is NOT_ACQUIRED and \MEA_2 is STATIC, NOT_ACQUIRED will be displayed.

– REQUESTEDFlagText = REQUESTED <hidden>

– REQUESTEDPriority = 4 <hidden>

– NOT_MAINTAINEDFlagText = NMAINT <hidden>

– NOT_MAINTAINEDPriority = 3 <hidden>

– NOT_ACQUIREDFlagText = NACQ <hidden>

– NOT_ACQUIREDPriority = 1 <hidden>

– NOT_RECEIVEDFlagText = NRCD <hidden>

– NOT_RECEIVEDPriority = 2 <hidden>

– INVALIDFlagText = INVAL <hidden>

– INVALIDPriority = 0 <hidden>

– ACQUIREDFlagText = <hidden>

– ACQUIREDPriority = 7 <hidden>

– DATA_INTERRUPTIONFlagText = INTERR <hidden>

– DATA_INTERRUPTIONPriority = 6 <hidden>

– STATICFlagText = STATIC <hidden>

– STATICPriority = 5 <hidden>

– ProcessingEnabledFlagText = <hidden>

– ProcessingDisabledFlagText = – <hidden>

– DISABLEDFlagText = <hidden>

– DISABLEDPriority = 5 <hidden>

– IN_LIMITSFlagText = <hidden>

– IN_LIMITSPriority = 7 <hidden>

– NOMINAL_LIMIT_VIOLATIONFlagText = <hidden>

– NOMINAL_LIMIT_VIOLATIONPriority = 4 <hidden>

– NOMINAL_LOW_LIMIT_VIOLATIONFlagText = <hidden>

– NOMINAL_LOW_LIMIT_VIOLATIONPriority = 2 <hidden>

– NOMINAL_HIGH_LIMIT_VIOLATIONFlagText = <hidden>

Copyright per DIN 34

- NOMINAL_HIGH_LIMIT_VIOLATIONPriority = 3 <hidden>
- DANGER_LOW_LIMIT_VIOLATIONFlagText = <hidden>
- DANGER_LOW_LIMIT_VIOLATIONPriority = 0 <hidden>
- DANGER_HIGH_LIMIT_VIOLATIONFlagText = <hidden>
- DANGER_HIGH_LIMIT_VIOLATIONPriority = 1 <hidden>
- UNDEFINEDFlagText = <hidden>
- UNDEFINEDPriority = 6 <hidden>
- EnableMonitoringFlag = False <hidden>

  Enable monitoring flags for standard elements.

- HideStatusWhenDisabled = False <hidden>

  If set, only the processing status is shown when processing is disabled. Acquisition status and monitoring status text are hidden.

- HideStatusWhenNotAcquired = False <hidden>

  If set, only the processing and acquisition status are shown when the item is not acquired. Monitoring status text is hidden.

- Resize_Always = False <hidden>

  If resize always is set, the window is always resized to the display parameter size as defined by GWDU when replacing a synoptic.

- RedrawAreas = True <hidden>

  If enabled, all dynamic object areas are redrawn after update to show static objects on top of these. Such static objects (e.g. a line crossing a bar chart) are hidden by the update procedure.

- HelpWindowWidth = 700 <hidden>

  Default window width of the synoptic display help.

- HelpWindowHeight = 400 <hidden>

  Default window height of the synoptic display help.

- RestoreOriginalColorWhenNotMonitored = no <hidden>

  Restores original color of output element when monitoring is disabled.

- PictureHistory = 50 <hidden>

  Maximum history depth of synoptic display’s forward/backward browse buffer

- LineWidth = 1 <hidden>

  Line width in pixels for line graphs in the range of 1..255.

+ Status_Display

- CCUItemWidth = 40 <hidden>

  Defines the size of the CCU item in character.

- CCUItemNextRow = Off <hidden>

  Defines if the CCU item shall be displayed on next row.

- CCUItem = On <hidden>

  Defines whether the CCU item shall be shown.

Copyright per DIN 34

```
  + Synoptic_Icons
    Attributes for synoptic display's small icons.
    – EmptyIconColor = gray <hidden>
      Color displayed if synoptic display slot is unused, i.e. no synoptic display is visible.
    – NoDataIconColor = white <hidden>
      Color displayed if corresponding synoptic display contains no data.
    – EmptySmallIcon = ${CGS_HOME}/etc/icons/empty.si <hidden>
      Icon image: synoptic display is not in use (empty).
    – No_DataSmallIcon = ${CGS_HOME}/etc/icons/no_data.si <hidden>
      Icon image: synoptic display has no data.
    – DISABLEDSmallIcon = ${CGS_HOME}/etc/icons/disabled.si <hidden>
    – IN_LIMITSSmallIcon = ${CGS_HOME}/etc/icons/in_limits.si <hidden>
    – SOFT_LIMIT_VIOLATIONSmallIcon = ${CGS_HOME}/etc/icons/soft_limit_violation.si <hidden>
    – DANGER_LIMIT_VIOLATIONSmallIcon = ${CGS_HOME}/etc/icons/danger_limit_violation.si <hidden>
    – UNDEFINEDSmallIcon = ${CGS_HOME}/etc/icons/undefined.si <hidden>
  + System_Advisory
    – UpdateFrequency = 5 <hidden>
      Defines how often the System Advisory shall be updated in seconds.
    – CheckFrequency = 15 <hidden>
      Defines how often it shall be checked that the System Advisory was updated.
    – AcknowledgeService = On <hidden>
      Determines if the acknowledge service is toggled on or off.
    – BeepOnWarning = true <hidden>
      Enables beep on warnings (yellow color)
  + Execution_Node_Status
    – MaxWindows = 4 <hidden>
      Defines the maximum number of Execution Node Status Displays that can be displayed in Parallel
    – AutoResize = True <hidden>
       Defines whether execution node status automatically resizes the window after selection of a new
Group.
  + Monitoring_Window
    – MaxWindows = 8 <hidden>
      Defines the maximum number of Monitoring Windows that can be displayed in parallel
    – DefaultDirectory = ${HOME}/.cgs/screen_setup_pool <hidden>
      Default directory for Load/Save file chooser.
  + Screen_Setup
```

Copyright per DIN 34

- AutoPositioningStatusDisplay = Yes <hidden>

  When set, the status display determines it position and size by itself instead of using the definition in the Screen setup.

- ConfigurationCheck = On <hidden>

  Defines whether a warning/confirmation is generated when a screen setup has to be loaded that was stored under a different configuration.

- LineWidth = 100_000 <hidden>

  Defines the maximum line width/length in a screen setup file.

- ListRows = 5 <hidden>

  Defines the number of visible rows of the screen setup files list.

- Directory = ${HOME}/.cgs/screen_setup_pool <hidden>

  Defines the location of the screen setup pool directory.

+ Validation

- Show = On <hidden>

  Controls display of the validation window.

- WindowWidth = 400 <hidden>

  Defines the size of the validation window.

- WindowHeight = 300 <hidden>

  Defines the size of the validation window.

- ErrorDiversion = On <hidden>

  Enables diversion of DataViews error messages to synoptic displays, default is on. If off, error messages are displayed on standard error (e.g. the console window).

+ Data_Control_Window

- WindowWidth = 330 <hidden>

  Defines the size of the data control window.

- WindowHeight = 40 <hidden>

  Defines the size of the data control window.

- KeyGotoFirst = 32602 <hidden>

  Key/action codes can be determined in OTC debug mode.
  In debug mode, all keys/actions on synoptic display (freeze mode) are printed to standard out.
  Function keys F1–F12 are 32605–32616, ASCII keys correspond to their code, e.g. space is 32. For a complete list, see XView Reference Manual.

- KeyGotoPrevious = 32598 <hidden>

  Key/action codes can be determined in OTC debug mode.
  In debug mode, all keys/actions on synoptic display (freeze mode) are printed to standard out.
  Function keys F1–F12 are 32605–32616, ASCII keys correspond to their code, e.g. space is 32. For a complete list, see XView Reference Manual.

- KeyGotoNext = 32600 <hidden>

  Key/action codes can be determined in OTC debug mode.
  In debug mode, all keys/actions on synoptic display (freeze mode) are printed to standard out.
  Function keys F1–F12 are 32605–32616, ASCII keys correspond to their code, e.g. space is 32. For a complete list, see XView Reference Manual.

Copyright per DIN 34

```
    - KeyGotoLast = 32596 <hidden>

      Key/action codes can be determined in OTC debug mode.
      In debug mode, all keys/actions on synoptic display (freeze mode) are printed to standard out.
       Function keys F1–F12 are 32605–32616, ASCII keys correspond to their code, e.g. space is 32. For a
complete list, see XView Reference Manual.

+ CIS

  + HLCL

    - LoginSequence = $HOME/.cgs/hlcl_login.seq <hidden>

  + Log

    - LoadErrorMessages = True <hidden>

    - LoadErrorMessagesFromDB = False <hidden>

    - CodeOffset = 5000 <hidden>

    - MessageFile = ${CGS_HOME}/etc/messages.def <hidden>

    - UserName = * <hidden>

  + Detailed_Info

    - WindowMemoryMaximum = 5_000 <hidden>

    - Padding_Spaces = 3 <hidden>

    - Not_Available_Text = <Not available> <hidden>

  + Global

    - BroadcastMessages = False <hidden>

      Determines to which Message Handler(s) all messages by CIS are sent:
      FALSE: only to local (CIS_nn) host,
      TRUE : broadcast to all hosts from System Topology.

  + Debug

    - EnableCORBAControl = False <hidden>

      Setting to TRUE enables external control of debug output for a
      running CIS process by a client for the 'Internal' CORBA interface.
      Since there is no access control for this interface, and debug output
      has impact on CIS performance and disk space consumption, this
      is a potential security risk and therefore disabled as default.

    - EnableExecutionLog = False <hidden>

      Control output of debugging information to file
        $CGS_HOME/local/tmp/vicos_cis.output.<hostname>.<date>:
      Setting to TRUE enables basic logging of CIS execution flow.
      For basic troubleshooting, expect about 100MB of output file/day.

    - EnableDetailedDump = False <hidden>

      Control output of debugging information to file
        $CGS_HOME/local/tmp/vicos_cis.output.<hostname>.<date>:
      Setting to TRUE enables extended output of internal and passed data.
      Useful for detailed troubleshooting, expect some 100MB of output file/day.

    - EnableExtensiveDDSDebug = False <hidden>

      Control output of debugging information to file
        $CGS_HOME/local/tmp/vicos_cis.output.<hostname>.<date>:
      Setting to TRUE enables detailed debug information from internal
      DDS (TM data) processing. Use only in special cases, may easily
      reach some GB of output file/day!
```

Copyright per DIN 34

```
  - KeepXMLReportFiles = False <hidden>

    Control handling of received XML report files from event log and
    command history in $CGS_HOME/data/*:
    Setting to TRUE will keep these temporary files for later inspection,
    instead of deleting them immediately. Note that this may quickly fill
    subdirectories of $CGS_HOME/data with thousands of *XML_Report* files!

+ ClientWatchdog

  - MaxClientReconnectRetries = 30 <hidden>

    Client Health Check Watchdog:
    Number of cycles to wait after a client has failed to accept
    SessionClient.Ping before client is declared dead and its
    session objects are destroyed. Timeout depends on number of
    clients and Max/MinClientPingDelay, default 30*10sec = 5min.
    A Session.Ping received from the client or a SessionClient.Ping
    by CIS accepted by the client will re–establish the client as healthy.

  - MaxClientPingDelay = 10000 <hidden>

    Client Health Check Watchdog:
    Maximum delay between two SessionClient.Ping requests sent
    by CIS to its clients (in milliSeconds), default 10_000 ms.
    Watchdog cycle time will be either MaxClientPingDelay or
    <number of clients> * MinClientPingDelay, whatever is larger.

  - MinClientPingDelay = 1000 <hidden>

    Client Health Check Watchdog:
    Minumum delay between two SessionClient.Ping requests sent
    by CIS to its clients (in milliSeconds), default 1_000 ms.
    Watchdog cycle time will be either MaxClientPingDelay or
    <number of clients> * MinClientPingDelay, whatever is larger.

+ HKTask

  - PollingCycle = 5000 <hidden>

    CIS Internal Housekeeping parameter acquisition:
    Polling interval for HK data acquisition
     (in milliSeconds), default 5_000 ms.
    In addition, some HK Parameter change reports may
    be triggered internally by asynchronuous events.

+ CorbaInterface

  - Enabled = True <hidden>

    Determines whether CIS Corba Interface shall be activated.
    When False, CIS will not provide the Corba Interface.

  - ServerPort = 7060 <hidden>

    TCP Port where CIS Corba interface listens to client requests.
    When changed, clients have to be adapted accordingly.

  - IIopUseHostnames = True <hidden>

    Defines how server is identified to clients in returned CORBA objects:
    TRUE  = by hostname   – default for CIS due to MCS requirements
    FALSE = by IP address – normal behaviour for OrbAda

  - POAAgentTaskStackSize = 655_360 <hidden>

    Do not change unless instructed by CGS Expert

+ SocketInterface

  - Enabled = False <hidden>
```

Copyright per DIN 34

```
      Determines whether CIS MOIS Socket Interface shall be activated.
      When False, CIS will not provide the Socket Interface.

    - ServerPort = 7008 <hidden>

      TCP Port where CIS MOIS Socket interface listens to client requests.
      When changed, clients have to be adapted accordingly.

    - CisUclLibrary = MOIS_CIS_LIB <hidden>

      Name of UCL User Library providing the procedures:
      CIS_Send_TC(Item:Issue_Item_Name(); Timeout:Duration; Status:Integer)
      CIS_Wait_For_Packet(ADU:ADU_Name; Timeout:Duration; Status:Integer).
      Without this library, Send_TC and Wait_for_ADU cannot be handled!

    - MaxNumberOfConnections = 20 <hidden>

      Maximum number of simultaneous MOIS socket client connections.

    - MaxNumberOfTMParameters = 100 <hidden>

      Maximum number of Items that can be acquired with a single TM data request.

    - AcquisitionTimeWithDate = False <hidden>

      Determines whether acquisition time of TM data is returned with date
      (as added in CS-RIBRE-ICD-0001 1/A) or without (only format in previous ICD).

+ TES

  This group contains all global configuration parameter for TES,
  TES_RPI, TES_API and the different groups.

  - COMMUNICATION [Administrator] = false <hidden>

    (A) Enables debug for tes communication protocol

        Range:                true/false
        Recommended value:    false

  - DEBUG [Administrator] = false <hidden>

    (A) Enables debug output in TES (if linked with debugging DEBUG_TES package).

        Range:                true/false
        Recommended value:    false

  - PERFORMANCE [Administrator] = false <hidden>

    (A) Enables logging of different time measueremente in TES.

        Range:                true/false
        Recommended value:    false

  - TRACE [Administrator] = false <hidden>

    (A) Enables logging of the special code in TES.

        Range:                true/false
        Recommended value:    false

  + API

    This group contains all TES_API specific configuration parameter.

    + CONTROLLER

      - STACK_SIZE [Administrator] = 1_000_000 <hidden>

        (A) Stack size of the api_controller task
```

Copyright per DIN 34

```
        Range:                  integer'range
        Recommended value:      1_000_000

    - TIMEOUT_PERIOD_FOR_READ_CMD [User] = 30.0 <hidden>

      (U) If an SAS does not call read_command within this time period, then
      TES will automatically disconnect the SAS

        Range:                  duration'range
        Recommended value:      30.0 [s]

    - TIMEOUT_FOR_READ [User] = 5_000 <hidden>

      (U) The timeout for communicating with TES for operations with answer back
          in [ms]

        Range:                  natural'range
        Recommended value:      5_000 [ms]

    - TIMEOUT_FOR_SEND [User] = 500 <hidden>

      (U) The timeout for sending data to TES (for ADU's and error messages)
          in [ms]

        Range:                  natural'range
        Recommended value:      500 [ms]

    - TIMEOUT_FOR_RESPONSE [User] = 2_000 <hidden>

      (U) The timeout while sending the acknowledges to TES through
          calls TES_API.ACKNOWLEDGE... in [ms]

        Range:                  natural'range
        Recommended value:      2_000 [ms]

    - DEFAULT_RETRIES [User] = 3 <hidden>

      (U) The number of retries when sending a message to TES

        Range:                  natural'range
        Recommended value:      3

    - KILL_OS_PROCESS [Administrator] = false <hidden>

      (A) For testing only.  If set to true then the TES_API will kill the
          OS process within which the SAS executes upon certain conditions, see
          parameters below

        Range:                  true/false
        Recommended value:      false

    - TIMEOUT_PERIOD_FOR_CONNECT [Administrator] = 40.0 <hidden>

      (A) For testing only.  If kill_os_process is set to true, then TES_API
          will kill the OS process if the SAS does not connect to TES within
          this period, seconds

        Range:                  duration'range
        Recommended value:      40.0 [s]

    - TIMEOUT_PERIOD_FOR_DISCONNECT [Administrator] = 40.0 <hidden>

      (A) For testing only.  If kill_os_process is set to true, then TES_API
          will kill the OS process if the SAS does not call disconnect after
          having received unload_application, seconds

        Range:                  duration'range
        Recommended value:      40.0 [s]

  + INTERNAL_MESSAGE_HANDLER

    - STACK_SIZE [Administrator] = 200_000 <hidden>
```

Copyright per DIN 34

```
      (A) Stack size of the api_internal_message_handler task

          Range:                integer'range
          Recommended value:    200_000

+ RPI

  This group contains all TES_RPI specific configuration parameter.

  - TIME_OUT_INIT [Administrator] = 300_000 <hidden>

    (A) Time-out for the INIT operation of the TES_RPI. It may be necessary
        to increase that value when loading very large databases. The time-out
        is expressed in milliseconds.

        Range:                integer'range
        Recommended value:    300_000 [ms]

  - WITH_TIME_OUT [Administrator] = true <hidden>

    (A) For testing only.  If set to false then the TES_RPI will not use
        time-outs for the communication with TES.

        Range:                true/false
        Recommended value:    true

  + CONTROLLER

    - STACK_SIZE [Administrator] = 150_000 <hidden>

      (A) Stack size of the rpi_controller task

          Range:                integer'range
          Recommended value:    150_000

  + REQUEST_HANDLER

    - STACK_SIZE [Administrator] = 200_000 <hidden>

      (A) Stack size of the rpi_request_handler task

          Range:                integer'range
          Recommended value:    150_000

+ KERNEL

  - DEFAULT_WORKSTATION [User] = HCI_01 <hidden>

    (U) Default workstation (HCI) for TES output in emergency case.
        Must be a logical name (see SYSTEM_TOPOLOGY_TABLE).

        Range:                array (1..255) of character
        Recommended value:    HCI_01

  - DELAY_WHEN_START_AP [Administrator] = 0.05 <hidden>

    (A) When starting APs as a monitoring action Kernel
        delays for this amount of time to let the AP start

        Range:                float'range
        Recommended value:    0.05 [s]

  + ADU

    + GENERATOR_POOL

      - INITIAL_NO_OF_AGENTS [Administrator] = 4 <hidden>

        (A) Initial number of adu_generator tasks, i.e. which are instantiated
        during elaboration
```

Copyright per DIN 34

```
            Range:                  natural'range
            Recommended value:      4

      - MAX_NO_OF_AGENTS [Administrator] = 100 <hidden>

        (A) The max number of adu_generator tasks, i.e. the max number of different
            adus that can be simulated in parallel

            Range:                  positive'range
            Recommended value:      100

    - MAX_NUMBER_OF_ADU_BUFFER [User] = 5 <hidden>

      (U) The maximum number of ADU buffers. In this buffer TES put complete
          ADUs on request.

          Range:                  integer'range
          Recommended value:      5

    + BUFFER

      - MAX_NUMBER_OF_ADUS [User] = 20 <hidden>

        (U) The maximum ADUs per ADU buffer. (see above)

            Range:                  integer'range
            Recommended value:      20

    + QUEUE

      - MAX_NUMBER_OF_QUEUED_ADUS [User] = 500 <hidden>

        (U) The maximum number of queued ADUs before TES starts to discard

            Range:                  integer'range
            Recommended value:      500

  + ARCHIVE

    - GENERATE_LOG_MESSAGE [User] = false <hidden>

      (U) Generate a message for each system library command
          'ENABLE_ARCHIVING' and 'DISABLE_ARCHIVING'

          Range:                  true/false
          Recommended value:      false

    - HIGH_ARCHIVE_CYCLE_LIMIT [Administrator] = 86399 <hidden>

      (A) Upper limit for change of archive file period, seconds

          Range:                  positve'range
          Recommended value:      86399 [s]

    - LIMIT_DISK_SPACE [User] = 20_000 <hidden>

      (U) Limit for issuing warning concerning the amount of free disk space

          Range:                  natural'range
          Recommended value:      20_000


      (U) Limit for issuing warning concerning the amount of free disk space

          Range:                  natural'range
          Recommended value:      20_000

    - LOW_ARCHIVE_CYCLE_LIMIT [Administrator] = 60 <hidden>

      (A) Lower limit for change of archive file period, seconds

          Range:                  positve'range
```

Copyright per DIN 34

```
            Recommended value:     60 [s]


          (A) Lower limit for change of archive file period, seconds

              Range:                 positve'range
              Recommended value:     60 [s]

      + DATA_PROCESSOR

        + ADU

          - CHECK_CHECKSUM [Administrator] = false <hidden>

            (A) allow or inhibit the checking of the checksum in ADU of type CCSDS
                packet or in response packet for software commanding. If it is set to
                true and the checksum is not detected to be
                correct, an error message is generated.

                Range:                 true/false
                Recommended value:     false

          - PROCESS_ON_INCORRECT_CHECKSUM [Administrator] = true <hidden>

            (A) allow or inhibit the processing of measurements contained in ADU of type
                CCSDS packets or the processing of response packets for software
                commanding if the checksum is incorrect.
                 This parameter is only used in the case where TES.KERNEL.DATA_PROCESSOR.ADU.CHECK_CHECKSUM
is
                set to true.

                Range:                 true/false
                Recommended value:     true

          - USE_APID_FOR_SEQUENCE_COUNT_CHECK [User] = false <hidden>

            (U) indicates which sequence count is used for internal processing
                and checking for telemetry packet order.

                true  – the ccsds packet sequence count in CCSDS primary header
                        is used for the internal processing. The check for
                        missing packets will be performed depend on apid.

                        In case the ADU is not a CCSDS packet, the Sequence_Number
                        from the T_ADU is always used.

                false – the Sequence_Number from the T_ADU is used for the internal
                        processing

                Range:                 true/false
                Recommended value:     false


          - USE_APID_FOR_TM_PACKET_IDENTIFICATION [User] = true <hidden>

            (U) indicates which source identification is used for internal processing
                for telemetry packets and transmission to HCI.

                All measurements, depend on telemetry packet (ADU), will inherit
                this information from the telemetry packet.

                true  – the ccsds packet application id in CCSDS primary header
                        (APID) is used for the packet identification

                        In case the ADU is not a CCSDS packet, the Source_Identifier
                        is set to "undefined source identifier".

                false – the Source_Identifier from the T_ADU is used for the packet
                        identification

                Range:                 true/false
                Recommended value:     true
```

Copyright per DIN 34

```
  – WARNING_ON_MAX_ADU_PACKET_LENGTH_VIOLATION [Administrator] = false <hidden>

    (A) allow or inhibit the checking of the packet length in ADU packets.
        If it is set to true and the actual length is grether than the
        in the MDB defined length, an error message will be generated.
        Actual length is for
          . CCSDS packets – the length written in length field of primary
            (PUS)          header.
        The check will be performed during processing of ADU (enabled by
        system library call ”start_acquisition”).

        Range:                true/false
        Recommended value:    false

  – WRITE_HK_VALUE [Administrator] = false <hidden>

    (A) Enables counting and writing of received and discarded ADUs

        Range:                true/false
        Recommended value:    false

– ADU_TIME_STAMP_FROM_CCSDS_PACKET [User] = false <hidden>

  (U) Indicates if the time stamp of the measurements (accessible via UCL)
      is based on the time stamp of the ADU–CCSDS_PACKET (true).

      Only valid if TES.KERNEL.DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_FROM_ADU (see above)
      is true and the ADU–CCSDS_PACKET has a Columbus/ISS secondary header or
                  the ADU–CCSDS_PACKET has an ATV–PUS secondary header!
      If the ADU–CCSDS_PACKET has no Columbus/ISS or ATV–PUS secondary header,
      the ADU time stamp is used for the measurements and no warning is
      reported.
      There is no effect if
      TES.KERNEL.DATA_PROCESSOR.MEASUREMENT_TIME_STAMP_FROM_ADU is false!

         Remember: All depend derived values are tagged with time stamp of
                   measurement too, when they are recalculated,
                   SMT and LT have the same time value.

      Range:                true/false
      Recommended value:    false

– DUMP_AFTER_INIT [Administrator] = false <hidden>

  (A) Enables dump of database contents after TES setup

      Range:                true/false
      Recommended value:    false

– ENABLE_ALL_ADUS_IN_REPLAY [Administrator] = false <hidden>

  (A) Enables all ADUs in the init–call

      Range:                true/false
      Recommended value:    false

– MEASUREMENT_TIME_STAMP_FROM_ADU [User] = false <hidden>

  (U) Indicates if the time stamp of the measurements (accessible via UCL)
      is based on ADU time stamp (true) or time of ADU processing in TES (false).
         Remember: All depend derived values are tagged with time stamp of
                   measurement too, when they are recalculated.

      Range:                true/false
      Recommended value:    false

– MESSAGE_ON_CONDITION [Administrator] = true <hidden>

  (A) allow or inhibit the generation of messages to the event log
      for each triggered condition.
```

Copyright per DIN 34

```
      Range:                 true/false
      Recommended value:     true

- MIN_TIMESLICE [Administrator] = 100 <hidden>

  (A) The minimum timeslice for calulation of cyclic derived values [ms].

      Range:                 100 .. natural'last
      Recommended value:     100 [ms]

- SHOW_ADU_RESUME_MESSAGE [Administrator] = false <hidden>

  (A) Enable/disable the message generation in case of ADU resumed
      to message handler log. Independ of this settings all messages
      will be logged to TRDB.

      Range:                 true/false
      Recommended value:     false

- SHOW_ADU_SUSPEND_MESSAGE [Administrator] = true <hidden>

  (A) Enable/disable the message generation in case of ADU suspend
      to message handler log. Independ of this settings all messages
      will be logged to TRDB.

      Range:                 true/false
      Recommended value:     true

- TIME_STAMP_IN_LT [User] = true <hidden>

  (U) Indicates if the
        a) time stamp of the measurements, software variables
           and derived values             (accessible via UCL/HLCL get_...)
        b) time stamp of gdu acknowledge  (accessible via UCL/HLCL issue)
        c) time stamp of adu packet       (accessible via UCL/HLCL
                                           get_full_packet, wait_for_adu...)
      is based on local time (true) or SMT (false).

      Range:                 true/false
      Recommended value:     true

- USE_LIMIT_SET_NUMBER_AS_CONDITION [Administrator] = false <hidden>

  (A) Allow or inhibit the usage of the MDB – defined Limit Set Number in
       enditem condition description as additional condition for the Action
       Type DISABLE_MONITORING only.

       For all other conditon actions this parameter has no effect!

       If this value is set to true, the condition is evaluated as follow:

           condition false => do nothing

           condition true  =>

              condition Action Type = DISABLE_MONITORING =>

                ACTION ITEM REFERENCE = measurement, variable, derived:

                  if condition Limit Set Number = current limit set
                  OR condition Limit Set Number = 0 => disable_monitoring

                ACTION ITEM REFERENCE = monitoring list, virtual pathname:

                  if condition Limit Set Number = current limit set (for at least
                                                  one single enditem)
                  OR condition Limit Set Number = 0 => disable_monitoring

                  else                              => do nothing (*)

              condition Action Type <> DISABLE_MONITORING => perform action
```

Copyright per DIN 34

```
          (*) In this case, the condition is not marked as triggered, because the
                           limit set check is part of the condition.

                  Remember: The condition action is performed once only as long as
                            the condition result is unchanged!

                Range:                  true/false
                Recommended value:      false

+ DDS

  - OOL_QUEUE_LENGTH [Administrator] = 100 <hidden>

    (A) size of the OOL_queue

        Range:                  integer'range
        Recommended value:      100

+ DEBUG

  - COMMAND_HISTORY [Administrator] = false <hidden>

    (A) Enables debug of command history calls

        Range:                  true/false
        Recommended value:      false

  - DATA_PROCESSOR_MONITORING [Administrator] = false <hidden>

    (A) Enables dump of enditem content after monitoring

        Range:                  true/false
        Recommended value:      false

  - DATA_PROCESSOR [Administrator] = false <hidden>

    (A) Enables debug output from data processor

        Range:                  true/false
        Recommended value:      false

  - LOCAL_DATABASE [Administrator] = false <hidden>

    (A) Enables debug output from local database during TES setup

        Range:                  true/false
        Recommended value:      false

  + STACK_MACHINE

    - DEBUG [Administrator] = false <hidden>

      (A) Enables debug output from I-code interpretation; one file per AP

          Range:                  true/false
          Recommended value:      false

    - DEBUG_FILE_NAME_PREFIX [Administrator] = ap <hidden>

      (A) Prefix of I-code interpretation debug output file name

          Range:                  array (1..255) of character
          Recommended value:      ap

    - WRITE_UCL_STMT [Administrator] = true <hidden>

      (A) Enables debug output from I-code interpretation; one file per AP

          Range:                  true/false
          Recommended value:      true

  + SYSTEM_LIBRARY
```

Copyright per DIN 34

```
    - HLCL_FILE_NAME [Administrator] = system_library.hlcl <hidden>

      (A) File name of HLCL debug output

          Range:                array (1..255) of character
          Recommended value:    system_library.hlcl

    - HLCL [Administrator] = false <hidden>

      (A) Enables debug output from HLCL–calls to the system library on file

          Range:                true/false
          Recommended value:    false

+ DISTRIBUTION_TABLE

  - MAX_NB_SID [Administrator] = 20_000 <hidden>

    (A) Maximal number of enditems in the test configuration.
        This number constrains a list that is used for remote
        operations on measurements / sw–variables, GDU's and GDU lists.

        Range:                integer'range
        Recommended value:    20_000

+ GDU

  + BUFFER

    - INITIAL_NUMBER [Administrator] = 5 <hidden>

      (A) Initial number of (pre–allocated) gdu_buffer tasks
          There will be 1 such task for each SAS that has the
          GDU service annonced.

          Range:                integer'range
          Recommended value:    5

  + CIS

    - LENGTH_FOR_CCSDS_PACKET_TYPE_CHANGE [Administrator] = 4089 <hidden>

      (A) Allow or inhibit the changing of CCSDS packet type to
          CCSDS_MEMORY_LOAD_PACKET (hex: B) in case of length violation.

          Remark: This config value will be evaluated for CCSDS packets
                  generation after execution of CIS procedures:

                       prepareSWOPCommand
                       prepareFLAPCommand
                       prepareTwoStageCommand

                  only, if the CIS procedures parameter
                  dataLoadPacketOverrideAllowed is set to true.

          If the config value is less than 4089 Byte, the CCSDS packet length
          will be checked agaist this value. If the CCSDS packet length is
          greater than the specified length, then the packet type will be
          automatic switched to CCSDS_MEMORY_LOAD_PACKET.

          A change message will be generated in case of changing only, if the
          CGS configuration file parameter TES.KERNEL.LOG.FORCED_VALUE_CHANGES is true.

          Range:                          0 .. 4089 [Byte]
          Recommended value:              4089 [Byte]

  + HANDLER

    - ISSUE_TIMEOUT_WHEN_MONITORING_EXCEPTION [User] = 5.0 <hidden>

      (U) Default timeout value when issuing gdus upon a monitoring
```

Copyright per DIN 34

```
          exception, seconds

          Range:                duration'range
          Recommended value:    5.0 [s]

    - TC_VERIFICATION_DISABLED [User] = false <hidden>

      (U) Default value for disable tc_verification to forbid
          the verification of all sended TC's independ of MDB definitions

          Range:                true/false
          Recommended value:    false

    - TC_PRECONDITION_CHECK_DISABLED [User] = false <hidden>

      (U) Default value for disable checking of tc preconditions. It forbid
          the check of preconditions of all TC's to be send independ of MDB
          definitions.

          Range:                true/false
          Recommended value:    false

    - USE_ONBOARD_PARAMETER_NUMBER [Administrator] = false <hidden>

      (A) Default value for using measurement onboard parameter number
          instead of sid for a pathname parameter in telecommands of type
          EGSE_PREDEFINED_TC, PUS_TC or BINARY_PACKETS. If it set to true,
          all pathname parameters in TC's are replaced by the
          corrosponding onboard parameter number. In case of no valid
          onboard parameter number (parameter number = 0), enditem is not a
          measurement or not known on local node, the TC will not be send
          and an error message will be generated!

          Range:                true/false
          Recommended value:    false

  - INITIAL_SEQUENCE_COUNT [Administrator] = 0 <hidden>

    (A) Specify the initial sequence count for CCSDS packets (the
        first value of a sequence count for any APID).

        Range:                0 .. 2047
        Recommended value:    0

  - MAX_CCSDS_PACKET_LENGTH [Administrator] = 4089 <hidden>

    (A) Allow or inhibit the checking of the packet length in GDU packets
        to be send. If the value is less than 4096 Byte, the GDU packet length
        will be checked agaist this value. GDU packet length means here for
          . CCSDS packets  – the length written in length field of primary
            (PUS/ATV)        header after end of parameter substitution or
                             after end of TC construction.
          . Binary packets – the length defined in mission database (MDB)
        An error message will be generated in case of violation
          (GDU packet length > GDU.MAX_type_PACKET_LENGTH < Range MAX).

        Range:                            0 .. 4089 [Byte]
        Range (Binary Packet):            1 .. 4096 [Byte]
        Recommended value:                4089 [Byte]
        Recommended value (Binary Packet): 4096 [Byte]

  - MAX_PUS_PACKET_LENGTH = 4089 <hidden>

  - MAX_ATV_PACKET_LENGTH = 4089 <hidden>

  - MAX_BINARY_PACKET_LENGTH = 4096 <hidden>

  - MIN_CCSDS_PACKET_LENGTH [Administrator] = 0 <hidden>

    (A) Allow or inhibit the checking of the packet length in GDU packets
        to be send. If the value is greater than 1 Byte, the GDU packet length
        will be checked agaist this value. GDU packet length means here for
```

Copyright per DIN 34

```
          . CCSDS packets  – the length written in length field of primary
            (PUS/ATV)        header after end of parameter substitution or
                             after end of TC construction.
          . Binary packets – the length defined in mission database (MDB)
        The length will be increased to the defined minimum packet length (null
        bytes added) in case of violation
          (GDU packet length < GDU.MIN_type_PACKET_LENGTH > Range MIN).

        Range:                            0 .. 4089 [Byte]
        Range (Binary Packet):            1 .. 4096 [Byte]
        Recommended value:                0 [Byte]
        Recommended value (Binary Packet): 1 [Byte]

  – MIN_PUS_PACKET_LENGTH = 0 <hidden>

  – MIN_ATV_PACKET_LENGTH = 0 <hidden>

  – MIN_BINARY_PACKET_LENGTH = 1 <hidden>

+ HK_VALUE_PROVIDER

  – UPDATE_CLOCK_PERIOD [User] = 1.0 <hidden>

    (U) Time period for cyclical update of the HK value for LT
        In replay it will be used to set the SMT in the HK data

        Range:                duration'range
        Recommended value:    1.0 [s]

  – UPDATE_DISK_PERIOD [User] = 30.0 <hidden>

    (U) Time period for cyclical update of the HK value for free disk space

        Range:                duration'range
        Recommended value:    30.0 [s]

  – UPDATE_DBS_PERIOD [User] = 60.0 <hidden>

    (U) Time period for cyclical update of the HK values from DBS

        Range:                duration'range
        Recommended value:    60.0 [s]

  – UPDATE_TSS_PERIOD [User] = 60.0 <hidden>

    (U) Time period for cyclical update of the HK values from TSS

        Range:                duration'range
        Recommended value:    60.0 [s]

+ LOG

  – IN_COMMAND_HISTORY [User] = false <hidden>

    (U) indicates whether all command relevant data are logged into the
        command history or not.
        true  – log into command history
        false – do not log into command history

        Range:                true/false
        Recommended value:    false

  – IN_LIMIT_AFTER_DANGER_EXCEPTION [User] = false <hidden>

    (U) Log re–entry messages after danger monitoring exception.

        Range:                true/false
        Recommended value:    false

  – IN_LIMIT_AFTER_NOMINAL_EXCEPTION [User] = false <hidden>

    (U) Log re–entry messages after nominal monitoring exception.
```

Copyright per DIN 34

```
        Range:                  true/false
        Recommended value:      false

  - IN_LIMIT_AFTER_DELTA_EXCEPTION [User] = false <hidden>

    (U) Log messages in re-entry case after delta monitoring exception.
        This parameter works in connection with at least one of the previous
        two parameters only. If IN_LIMIT_AFTER_DANGER_EXCEPTION and
        IN_LIMIT_AFTER_NOMINAL_EXCEPTION set to false, this parameter is without
        meaning.

        Range:                  true/false
        Recommended value:      false

  - FORCED_VALUE_CHANGES [User] = true <hidden>

    (U) Log messages for item value changes that are forced by TES config
        parameters (e.g. TES.KERNEL.GDU.MIN_CCSDS_PACKET_LENGTH).

        Range:                  true/false
        Recommended value:      true

  - CALIBRATION_ERROR_ONCE [User] = true <hidden>

    (U) Log messages in case of calibration error

            once per measurement until a calibration was successful  (true)
        or
            one message per one calibration error                    (false).

        Range:                  true/false
        Recommended value:      true

  - MAX_QUEUED_MESSAGES [Administrator] = 50 <hidden>

    (A) Max number of messages that can be queued for logging before the log
        object forces logging to be done

        Range:                  natural'range
        Recommended value:      50

  - VALUE_CHANGES [User] = true <hidden>

    (U) Log exception messages for value changes for items
        which are enabled for value change monitoring.

        Range:                  true/false
        Recommended value:      true

+ MONITOR

  - ACTION_HANDLERS [Administrator] = 5 <hidden>

    (A) Number of action_handlers in the monitor, i.e. the number of monitor
        actions (GDUs or APs) that can be handled in parallel

        Range:                  integer'range
        Recommended value:      5

+ PI_CONTROLLER

  - POLLING_INTERVAL [Administrator] = 10 <hidden>

    (A) Polling interval for General Comms passed to
        general_comms.setup_communication

        Range:                  poitive'range
        Recommended value:      10

+ REPLAYER
```

Copyright per DIN 34

```
  - USE_REPLAY_TIME [User] = false <hidden>

    (U) Indicates if the log events / messages produced during a replay session
        use the replay local time or the current local time.

        Range:                true/false
        Recommended value:    false

  - SMT_UPDATE_PERIOD_IN_REPLAY [Administrator] = 5.0 <hidden>

    (A) The frequency to update the SMT clock in replay mode

        Range:                duration'range
        Recommended value:    5.0 [s]

  - GDU_MESSAGES [User] = true <hidden>

    (U) Enables the generation of an error message for each GDU that is replayed
        from the archive files. This has no impact on messages issued when
        generating a GDU from the replay sesion (e.g. as response of a monitoring
        exception or from an AP).
        This should be set to false when replaying sessions with a lot of GDU's,
        especially when the replay is accelerated.

        Range:                true/false
        Recommended value:    true

+ REQUEST_FETCHER

  - NO_OF_MESSAGE_AGENTS [Administrator] = 10 <hidden>

    (A) The number of message_agent tasks, i.e. the max number of requests from
        TES_RPI or TES_API being served in TES_Core in parallel

        Range:                natural'range
        Recommended value:    10

  - WAITING_PERIOD_AFTER_FAILURE [Administrator] = 10.0 <hidden>

    (A) The period in seconds that TES_Core waits after having received an
        error from the communication SW before trying to call again

        Range:                duration'range
        Recommended value:    10.0 [s]


    (A) The period in seconds that TES_Core waits after having received an
        error from the communication SW before trying to call again

        Range:                duration'range
        Recommended value:    10.0 [s]

+ SAS

  - ENABLE_ACQUISITION_TIMEOUT [User] = 60.0 <hidden>

    (U) Timeout for ack from SAS upon enable ADU request, seconds.

        Range:                float'range
        Recommended value:    60.0 [s]

  - DISABLE_ACQUISITION_TIMEOUT [User] = 60.0 <hidden>

    (U) Timeout for ack from SAS upon disable ADU request, seconds

        Range:                float'range
        Recommended value:    60.0 [s]

  - LOAD_APPLICATION_TIMEOUT [User] = 30.0 <hidden>

    (U) Timeout for connect from SAS after being started, seconds
```

Copyright per DIN 34

```
      Range:                float'range
      Recommended value:    30.0 [s]

- INIT_APPLICATION_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon init application, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- UNLOAD_APPLICATION_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon unload application, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- RESET_APPLICATION_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon reset application, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- GET_APPLICATION_STATUS_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon get application status, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- WRITE_MESSAGE_TO_APPLICATION_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon write message to application, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- DOWNLOAD_FILE_TO_APPLICATION_TIMEOUT [User] = 60_000 <hidden>

  (U) Timeout for ack from SAS upon download file to application, milliseconds

      Range:                positive'range
      Recommended value:    60_000 [ms]

- INITIALISE_TIMEOUT [User] = 30_000 <hidden>

  (U) Timeout for ack from another TES in initialisation
      exchange of distribution tables), milliseconds

      Range:                positive'range
      Recommended value:    30_000 [ms]

+ SAS_STARTER

  - LOAD_APPLICATION [Administrator] = true <hidden>

    (A) For testing only.  Avoids that an SAS is started as an OS process,
        thus the SAS can be started manually

        Range:                true/false
        Recommended value:    true

  - WITH_OUTPUT [Administrator] = false <hidden>

    (A) For testing only.  Allow writing of all SAS output into a file
        located in $CGS_HOME/local/tmp.

        Range:                true/false
        Recommended value:    true

- START_APPLICATION_TIMEOUT [User] = 60_000 <hidden>
```

Copyright per DIN 34

```
    (U) Timeout for ack from SAS upon start application, milliseconds

        Range:                positive'range
        Recommended value:    60_000 [ms]

+ STACK_SIZE

  - ADU_AGENT [Administrator] = 500_000 <hidden>

    (A) Stack size of the adu_agent task

        Range:                integer'range
        Recommended value:    500_000

  - ADU_GENERATOR [Administrator] = 100_000 <hidden>

    (A) Stack size of the adu_generator tasks

        Range:                integer'range
        Recommended value:    100_000

  - COMMAND_ROUTER [Administrator] = 100_000 <hidden>

    (A) Stack size of the command_router task

        Range:                integer'range
        Recommended value:    100_000

  - AP_COMMUNICATOR [Administrator] = 20_000 <hidden>

    (A) Stack size of the ap_communicator tasks

        Range:                integer'range
        Recommended value:    20_000

  - AP_CONTROLLER [Administrator] = 500_000 <hidden>

    (A) Stack size of the ap_controller tasks

        Range:                integer'range
        Recommended value:    500_000

  - GDU_BUFFER [Administrator] = 200_000 <hidden>

    (A) Stack size of the gdu_buffer tasks

        Range:                integer'range
        Recommended value:    200_000

  - ARCHIVE [Administrator] = 100_000 <hidden>

    (A) Stack size of the archive task

        Range:                integer'range
        Recommended value:    100_000

  - LOG [Administrator] = 220_000 <hidden>

    (A) Stack size of the log task

        Range:                integer'range
        Recommended value:    220_000

  - CLOCK_CONTROLLER [Administrator] = 50_000 <hidden>

    (A) Stack size of the clock_controller task

        Range:                integer'range
        Recommended value:    50_000

  - DDS_BUFFER [Administrator] = 100_000 <hidden>
```

Copyright per DIN 34

```
  (A) Stack size of the dds_buffer task

      Range:                integer'range
      Recommended value:    100_000

- MODE_SWITCH_DETECTOR [Administrator] = 100_000 <hidden>

  (A) Stack size of the mode_switch_detector task

      Range:                integer'range
      Recommended value:    100_000

- HK_VALUE_PROVIDER [Administrator] = 270_000 <hidden>

  (A) Stack size of the hk_value_provider tasks

      Range:                integer'range
      Recommended value:    270_000

- FILE_ARCHIVER [Administrator] = 250_000 <hidden>

  (A) Stack size of the file archive task

      Range:                integer'range
      Recommended value:    250_000

- REQUEST_FETCHER [Administrator] = 200_000 <hidden>

  (A) Stack size of the request_fetcher task

      Range:                positive'range
      Recommended value:    200_000

- MESSAGE_AGENT [Administrator] = 500_000 <hidden>

  (A) Stack size of the message_agents tasks

      Range:                integer'range
      Recommended value:    500_000

- PI_CONTROLLER [Administrator] = 200_000 <hidden>

  (A) Stack size of the pi_controller task

      Range:                integer'range
      Recommended value:    200_000

- REPLAYER [Administrator] = 300_000 <hidden>

  (A) Stack size of the replayer task

      Range:                integer'range
      Recommended value:    300_000

- SW_CMDER [Administrator] = 300_000 <hidden>

  (A) Stack size of the replayer task

      Range:                integer'range
      Recommended value:    300_000

- TES_CON_EXEC [Administrator] = 1_000_000 <hidden>

  (A) Stack size of the tes_con_exec task

      Range:                integer'range
      Recommended value:    1_000_000

- PACKET_STORAGE [Administrator] = 220_000 <hidden>

  (A) Stack size of the packet_storage task
```

Copyright per DIN 34

```
        Range:                  integer'range
        Recommended value:      220_000

  - SAS_STARTER [Administrator] = 100_000 <hidden>

    (A) Stack size of the sas_starter task

        Range:                  integer'range
        Recommended value:      100_000

  - UCL_INTERPRETER [Administrator] = 512 <hidden>

    (A) Size of Interpreter

        Range:                  Positive'Range
        Recommended value:      512

  - UCLI_CONTROLLER [Administrator] = 220_000 <hidden>

    (A) Stack size of the ucli_controller task

        Range:                  integer'range
        Recommended value:      220_000

  - MONITOR_ACTION_HANDLER [Administrator] = 300_000 <hidden>

    (A) Stack size of the action_handlers tasks

        Range:                  integer'range
        Recommended value:      300_000

+ ACTION_BUFFER

  - BUFFER_LENGTH [Administrator] = 100 <hidden>

    (A) The length of the action_buffer data structure

        Range:                  integer'range
        Recommended value:      100

+ SW_CMDER

  - MAX_PARALLEL_SWOPS [Administrator] = 50 <hidden>

    (A) Maximum number of SW commands handled at the same time

        Range:                  integer'range
        Recommended value:      50

  - EXEC_FLAP_SWOP_PATHNAME [User] =  <hidden>

    (U) Pathname for the SW command to be used for starting a FLAP in
        the onboard system without waiting for the FLAP to terminate.

        Range:                  array (1..255) of character
        Recommended value:      No

  - EXECWAIT_FLAP_SWOP_PATHNAME [User] =  <hidden>

    (U) Pathname for the SW command to be used for starting a FLAP in
        the onboard system with subsequent waiting for the FLAP to terminate

        Range:                  array (1..255) of character
        Recommended value:      No

  - ISSUE_TIMEOUT_WHEN_NO_DELAY [User] = 5.0 <hidden>

    (U) Default timeout for telecommand part value when issuing a software
        command with null delay, seconds

        Range:                  duration'range
```

Copyright per DIN 34

```
            Recommended value:     5.0 [s]

      – USE_FLAP_OB_PARAMETER_FORMAT [Administrator] = true <hidden>

        (A) Use the parameter encoding schema for FLAP parameters defined in
            DMS ICD (COL–RIBRE–ICD–0065.2 7/– 12.06.2003) = true,

            or

            UCL Virtual Stack Machine and I Code Reference Manual
            (CGS–RIBRE–STD–0003 1/– 01.02.2002)           = false
            (new time format and duration as two word scalar)

            Range:                 true/false
            Recommended value:     true

    + UCLI_CONTROLLER

      – HK_VALUE_UPDATE_PERIOD [Administrator] = 5.0 <hidden>

        (A) Period for writing the ”current UCL statement” HK value, seconds

            Range:                 duration’range
            Recommended value:     5.0 [s]

      – TIME_OUT_ON_CONFIRM [Administrator] = 60.0 <hidden>

        (A) Upon forced stop of TES, when APs are running, if something goes wrong
            so that the APs do not stop, then this is the timeout in seconds

            Range:                 duration’range
            Recommended value:     60.0 [s]

      – REDELIVERY_DELAY [Administrator] = 0.02 <hidden>

        (A) When a wait event could not immediately be forwarded to the client,
            duration to wait before retrying

            Range:                 duration’range
            Recommended value:     0.02 [s]

      – NUMBER_OF_UCL_INTERPRETER [Administrator] = 20 <hidden>

        (A) number of ap interpreters. This determines the number of AP that
            can run in parallel.

            Range:                 1 .. 40
            Recommended value:     20

      – MAXIMUM_NUMBER_OF_EMERGENCY_APS [Administrator] = 5 <hidden>

        (A) The number of slots that shall be reserved for emergency APs

            Range:                 1..TES.KERNEL.UCLI_CONTROLLER.NUMBER_OF_UCL_INTERPRETER
            Recommended value:     5

    + USER_REGISTER

      – MAX_NO_OF_USERS [Administrator] = 32 <hidden>

        (A) Max number of HCIs/TSCVs that can be connected to TES

            Range:                 integer’range
            Recommended value:     32

+ Tscv

  + Stack_Size

    – Mainloop = 1000 <hidden>

    – Await_Quit = 32 <hidden>
```

Copyright per DIN 34

```
    - L_Tcs_Controller = 1000 <hidden>

    - L_Tcs_Session_Mgt = 1100 <hidden>

    - L_Tcs_Status_Manager = 1500 <hidden>

    - Alive = 250 <hidden>

    - Close_Session = 300 <hidden>

    - Debug_Output = 1400 <hidden>

    - Debug_Semaphor = 1200 <hidden>

  + Internal

    - Tss_Stop_Signal = SIGUSR2 <hidden>

      (E) TSS_STOP is called by TSS_EXTIF package if this signal is received.

    - Lock_Update_Interval = 5 <hidden>

      (E) Update interval of TSCV lock file.

    - Seconds_Delay_During_Tes_Init = 3 <hidden>

      (E) Interval between each time TES status is updated during initiation.

    - Pcs_Timeout = 5.0 <hidden>

      (E) Timeout for process creation server to start an application from TSCV.

    - Tsp_Launch_Wait_Time = 20 <hidden>

      (E) The maximum time TSCV will wait for response after having launched a TSP.

    - Size_Of_Debug_Buffer = 500 <hidden>

      (E) Number of 80 characters lines buffered for output application debug output.

    - Debug_Tracing_Is_On = FALSE <hidden>

      (E) Whether or not to output debug tracing.

    - Debug_Trace_Only_To_File = TRUE <hidden>

      (E) Whether or not the debug trace output should be printed also to the screen.

    - Debug_Trace_Filename = tscv_debug_tscv.trace <hidden>

      (E) Name of debug file (directory is $CGS_HOME/local/tmp).

  + User

    - Status_Update_Interval = 5 <hidden>

       (U) Update interval for getting status of TES/HCI/DBS nodes. A value of zero will disable the
cyclic checking.

    - Test_Session_Name_Format = %user%.%prefix%.%year%%month%%day%:%hour%:%minute%.%second% <hidden>

      (U) The format string of a new created test session.
      Each parameter enclosed by the characters '%' will be replaced by TSCV. Available parameters are:
          user   : the user name
          prefix : typed by the user at run time
          year   : currently year
          month  : currently month
          day    : currently day
          hour   : currently hour
          minute : currently minute
          second : currently Second
```

Copyright per DIN 34

- Batch_Session_Extension = FALSE <hidden>

  (U) Extends session name (User, Time) also for batch mode.

- Show_Generate_Window = TRUE <hidden>

  Using of generate window (generate SCOE files, CLS batch compile) at TC startup.

- Show_Ccu_Internal_Id = FALSE <hidden>

  Showing the CCU internal ID in certain windows.

+ TSS

This group contains all global configuration parameter for TSS
(Time Synchronisation Software).

- Ntp_Status_File [Administrator] = /usr/tmp/ntp_status <hidden>

  (A) Denotes the name of the ntp status file.

      Type: String
      Recommended value: /usr/tmp/ntp_status

- Shared_Mem_Key_File [Administrator] = /tss_shmkeyfile <hidden>

  (A) Filename for shared memory key. Relative to $CGS_HOME/etc.

      Type: String
      Recommended value: /tss_shmkeyfile

- Semaphore_Key_File [Administrator] = /tss_semkeyfile <hidden>

  (A) Filename for semaphore key. Relative to $CGS_HOME/etc

      Type: String
      Recommended value: /tss_semkeyfile

- Smt_Monitor_Delay [Administrator] = 0.1 <hidden>

  (A) Maximum accuracy of system library call DELAY.

      Type: Duration
      Recommended value: 0.1

- Update_Rate_Of_Ntp_States_In_Shm [Administrator] = 10 <hidden>

  (A) The update rate of the NTP state in the shared memory.

      Type: Positive
      Recommended value: 10

- Local_Host_Check_Timeout [Administrator] = 0.05 <hidden>

  (A) Internal delay used for hostname detection.

      Type: Duration
      Recommended value: 0.05

+ TEV

This group contains the configuration parameters specific to TEV.

- ARCHIVE_FILES_SIZE_LIMIT = 30 <hidden>

  Limit size for archive files before warning

- LACIS_UNDEFINED_VALUE = –1.E30 <hidden>

  Default value to print into data listing results in Lacis format if there is nothing read

Copyright per DIN 34

```
– CSV_COLUMN_SEPARATOR = ; <hidden>
  Column separator to be used in result files in CSV format
– CSV_DECIMAL_SEPARATOR = , <hidden>
  Decimal separator to be used in result files in CSV format
– LT_CONTINUOUS = TRUE <hidden>
  Specifies whether LT should be regarded as continuous
– SMT_CONTINUOUS = TRUE <hidden>
  Specifies whether SMT should be regarded as continuous
+ STACK_SIZES
  This group contains the various stack sizes specific to TEV.
  – TASK_TO_PERFORM_RESULT = 1_300_000 <hidden>
    Specifies stack size for result task
  – TASK_TO_PERFORM_DEFINITION = 300_000 <hidden>
    Specifies stack size for definition task
  – TASK_TO_QUIT_TEV = 50_000 <hidden>
    Specifies stack size for quit task
  – MAIN_LOOP_TASK = 400_000 <hidden>
    Specifies stack size for main loop task
  – MAIN_CONTROLLER_TASK = 150_000 <hidden>
    Specifies stack size for main controller task
  – WD_FILES_TASK = 300_000 <hidden>
    Specifies stack size for WD files task
  – TIME_FRAME_TASK = 1_500_000 <hidden>
    Specifies stack size for time frame task
  – EXECUTION_SESSIONS_TASK = 300_000 <hidden>
    Specifies stack size for execution sessions task
  – EVALUATION_SESSIONS_TASK = 300_000 <hidden>
    Specifies stack size for evaluation sessions task
  – ERROR_HANDLER_TASK = 100_000 <hidden>
    Specifies stack size for error handler task
  – PARAMETERS_TASK = 1_500_000 <hidden>
    Specifies stack size for parameters task
  – DATA_SET_TASK = 1_450_000 <hidden>
    Specifies stack size for data set task
  – DUMP_ACCESS_DBS_TASK = 1_200_000 <hidden>
    Specifies stack size for dump access DBS task
```

Copyright per DIN 34

```
    – MERGE_TASK = 300_000 <hidden>

      Specifies stack size for merge task

    – BATCH_DRIVER_TASK = 400_000 <hidden>

      Specifies stack size for batch driver task

    – SAS_TOOL_TASK = 300_000 <hidden>

      Specifies stack size for SAS tool task

    – IMPORT_EXPORT_TASK = 400_000 <hidden>

      Specifies stack size for import–export task

    – FINAL_ARCHIVE_TASK = 400_000 <hidden>

      Specifies stack size for final archive task

– Version = @(#) cgs_configuration_defaults.xml cgs_6.3/41  01/08/07 13:59:42@(#)

  CGS internal configuration version in the configuration management system.
```

- **TEV Syntax for Ascii Definition for Data Set Generation: data_set.def_syntax**

TEV_DEFINITION

TIME_FRAME_BEGIN	<DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END	<DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST	<STRING(80)>
USER_EVENT_LAST	<STRING(80)>
SELECT_ON	<LT | SMT>
ORDER_BY	<LT | SMT>
SAMPLING_EVERY	<NATURAL>
— only for sampling mode N_SAMPLING	
SAMPLING_TIME_INTERVAL	<HH:MM:SS.MSS>
— only for sampling mode TIME_BASED_SAMPLING	
SAMPLING_ALLOWED_ERROR	<HH:MM:SS.MSS>
— only for sampling mode TIME_BASED_SAMPLING	
PARAMETER_SET	# maximum 4 parameter sets
SOURCE_TYPE	<ARCHIVE | EVL>
TEST_NODE	<STRING(20)>
SELECTED_PARAMETERS	# maximum 50 parameters overall !
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_SET	# maximum 4 parameter sets
SOURCE_TYPE	<ARCHIVE | EVL>
TEST_NODE	<STRING(20)>

Copyright per DIN 34

```
SELECTED_PARAMETERS             # maximum 50 parameters overall !
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>

PARAMETER_SET                   # maximum 4 parameter sets
SOURCE_TYPE                     <ARCHIVE | EVL>
TEST_NODE                       <STRING(20)>
SELECTED_PARAMETERS             # maximum 50 parameters overall !
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>

PARAMETER_SET                   # maximum 4 parameter sets
SOURCE_TYPE                     <ARCHIVE | EVL>
TEST_NODE                       <STRING(20)>
SELECTED_PARAMETERS             # maximum 50 parameters overall !
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
PARAMETER_NAME                  <STRING(256)>
```

- Syntax for Ascii definition for data listing generation : data_list.def_syntax

```
TEV_DEFINITION

# Use ASCII format for generation of ascii text listing
# Use ADT   format for generation of a binary file, which can be
# read using tev_api library
# Use EXCEL format to generate a file using the comma separated value format
# The option GENERATE_FORMAT will be used only if no option –fo is given
# through the start_tev batch command line
# Optional
```

Copyright per DIN 34

```
# Default uses :  ASCII
GENERATE_FORMAT                         <ASCII | ADT  | EXCEL | LACIS >
# Use WITH_RAW_VALUES YES to force display of the raw values on the next line
# after the engineering values line
# Optional
# Default uses :  NO
WITH_RAW_VALUES                         <NO | YES>
# Enter a string to specify a title for the Data Listing
# Optional
# Default uses : (no string)
LISTING_TITLE                           <STRING(110)>
# Time frame for extracting the data
# Optional
# Default uses : time frame of the data set
TIME_FRAME_BEGIN                        <DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END                          <DD.MM.YYYY HH:MM:SS.MSS>
# Name of the data set from which the data shall be
# extracted
# Mandatory
# No default
DATA_SET_NAME                           <STRING(20)>
# Source of the data set :
# WORKING_DIR : the data set is in the $HOME/wd/tev/RESUTS/DATA_SET directory
# TRDB        : the data set is in the TRDB (under an evaluation session directory)
# Optional
# Default uses : WORKING_DIR
DATA_SET_SOURCE                         <WORKING_DIR | TRDB>


# Measurements for which TEV shall generate the listing
# Mandatory : at least one measurement
# No default
SELECTED_PARAMETERS       # maximum 50 measurements for EXCEL format resp. 10 parameters for ASCII and ADT
format
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
PARAMETER_NAME                          <STRING(256)>
```

Copyright per DIN 34

PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>
PARAMETER_NAME	<STRING(256)>

- Syntax for Ascii definition for events listing : event_list.def_syntax

TEV_DEFINITION

SELECT_ON	<LT | SMT>
ORDER_BY	<LT | SMT>
DISPLAY_EXTRA_TEXT	<NO | YES>
DISPLAY_LONG_TEXT	<NO | YES>
GENERATE_FORMAT	<ASCII | ADT>
NORMAL_CRITERION	
TIME_FRAME_BEGIN	<DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END	<DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST	<STRING(80)>
USER_EVENT_LAST	<STRING(80)>
PRODUCER	<STRING(256)>
TYPE	<STRING(256)>
GROUP	<STRING(256)>
SHORT_TEXT	<STRING(256)>
EXTRA_TEXT	<STRING(256)>
LONG_TEXT	<STRING(256)>
NORMAL_CRITERION	
TIME_FRAME_BEGIN	<DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END	<DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST	<STRING(80)>
USER_EVENT_LAST	<STRING(80)>
PRODUCER	<STRING(256)>
TYPE	<STRING(256)>
GROUP	<STRING(256)>
SHORT_TEXT	<STRING(256)>

Copyright per DIN 34

```
EXTRA_TEXT                          <STRING(256)>
LONG_TEXT                           <STRING(256)>

NORMAL_CRITERION
TIME_FRAME_BEGIN                    <DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END                      <DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST                    <STRING(80)>
USER_EVENT_LAST                     <STRING(80)>
PRODUCER                            <STRING(256)>
TYPE                                <STRING(256)>
GROUP                               <STRING(256)>
SHORT_TEXT                          <STRING(256)>
EXTRA_TEXT                          <STRING(256)>
LONG_TEXT                           <STRING(256)>

NORMAL_CRITERION
TIME_FRAME_BEGIN                    <DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END                      <DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST                    <STRING(80)>
USER_EVENT_LAST                     <STRING(80)>
PRODUCER                            <STRING(256)>
TYPE                                <STRING(256)>
GROUP                               <STRING(256)>
SHORT_TEXT                          <STRING(256)>
EXTRA_TEXT                          <STRING(256)>
LONG_TEXT                           <STRING(256)>
```

- Syntax for Ascii definition for graph generation in one shot : one_shot_graph.def_syntax

```
TEV_DEFINITION

# Default for the time frame : the time frame of the session(s).
TIME_FRAME_BEGIN                    <DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END                      <DD.MM.YYYY HH:MM:SS.MSS>
# Default : LT
SELECT_ON                           <LT | SMT>
# Default : LT
ORDER_BY                            <LT | SMT>

# Default : empty strings
GRAPH_NAME                          <STRING(50)>
USER_EVENT_FIRST                    <STRING(80)>
USER_EVENT_LAST                     <STRING(80)>

# Default : N sampling : 1
SAMPLING_EVERY                      <NATURAL>
SAMPLING_TIME_INTERVAL              <STRING(12)> — time format DELTA_T
SAMPLING_ALLOWED_ERROR              <STRING(12)> — time format EPSILON
# With_Raw_Values is only useful for graph types LINE. In this case only one selected parameter
# is allowed. Default : NO
WITH_RAW_VALUES                     <NO | YES>

# Default : Line
GRAPH_TYPE                          <LINE | BAR >

# Default : empty strings
X_AXIS_NAME                         <STRING(50)>
Y_AXIS_NAME                         <STRING(50)>
```

Copyright per DIN 34

```
X_AXIS_SCALING_MIN                  <STRING(10)>
# only useful for graph type XY
X_AXIS_SCALING_MAX                  <STRING(10)>
# only useful for graph type XY
Y_AXIS_SCALING_MIN                  <STRING(10)>
Y_AXIS_SCALING_MAX                  <STRING(10)>
# Default : NO
X_REL_TIME                          <NO | YES> # only useful for graph type LINE

PARAMETER_SET
# Source_Type : the type of the files from which the data are extracted.
# Archive files : ARCHIVE or Engineering Value Logbook : EVL. Default : ARCHIVE
SOURCE_TYPE                         <ARCHIVE | EVL>
# Test_Node is the application name , as defined in the topology table , which has
# produced the data. This entry is mandatory
TEST_NODE                           <STRING(20)>

# Selected_Parameters : up to 5 measurements for graph type LINE, up to 2 measurements for XY graph type
# Only 1 measurement when WITH_RAW_VALUES : TRUE
# This enry is mandatory
SELECTED_PARAMETERS

# At least one PARAMETER_NAME is mandatory
# The six next entries described each measurement. Only Parameter_Name is mandatory for one measurement.
PARAMETER_NAME                      <STRING(256)>
X_PARAMETER_NAME                    <STRING(256)> # only useful for graph type XY
Y_PARAMETER_NAME                    <STRING(256)> # only useful for graph type XY
COLOUR                              <BLACK | RED | GREEN | BLUE | YELLOW | MAGENTA | WHITE |
BROWN | CYAN | ORANGE | GREY>
LINE_STYLE                          <SOLID | DASHED | DOTTED | DASH_DOT | LONG_DASH>
LINE_THICKNESS                      <0 | 1 | 2 | 4>
```

- Syntax for Ascii definition for raw data dump generation : raw_data_dump.def_syntax

```
TEV_DEFINITION

PRODUCER                            <STRING(20)>
TIME_FRAME_BEGIN                    <DD.MM.YYYY HH:MM:SS.MSS>
TIME_FRAME_END                      <DD.MM.YYYY HH:MM:SS.MSS>
USER_EVENT_FIRST                    <STRING(80)>
USER_EVENT_LAST                     <STRING(80)>
SELECT_ON                           <LT | SMT>
ORDER_BY                            <LT | SMT>
GENERATE_FORMAT                     <ASCII | ADT>
OUTPUT_FORMAT                       <HEXADECIMAL | DECIMAL | ASCII>
OUTPUT_BYTES_PER_LINE               <POSITIVE (DECIMAL: 8 | 16 | 24, HEXADECIMAL: 8 | 16 | 24 | 32,
ASCII: 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64)>
OUTPUT_SUMMARY_FORMAT               <YES | NO>

SELECTED_ADUS                       # maximum 20 ADUs (given by pathname or nickname)
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
ADU_NAME                            <STRING(256)>
```

Copyright per DIN 34

ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
ADU_NAME	<STRING(256)>
SELECTED_GDUS	# maximum 20 GDUs (given by pathname or nickname)
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>
GDU_NAME	<STRING(256)>

Copyright per DIN 34

L CGS SCREEN SETUP

L-1 Screen Setups and Window Definitions

There are several means to control and predefine the setup of the workstation's screens for CGS use.

The following save function are implemented separately:

Position and size of **Task Selector, Info, DB Selector, Welcome and Message Handler** can be saved by the Utilities –> Save_Workspace function of the window manager.

Message Handler window contents can be saved by storing the properties of the message window.

HCI Windows can be saved by the built–in Screen–Setup_Maintenance function The screen definition files are stored under $HOME/.cgs/screen_setup_pool.

TEV Windows cannot be saved. A setup can, however, be defined in the $CGS_HOME/etc/XDefaults file.

For each TEV window (main windows and tool windows), the position of the window can be specified within this file. The file is attached to the X Resource database before TEV is started.

TSCV Windows cannot be saved

The size and position of the main window of TSCV can be given as parameters to the startup command.

The user may optionally specify by command line parameters the placement of the main TSCV window. If the command line contains the tag **–geometry**, tscv will expect to find integer values for the width, height, horizontal postion, vertical position, respectively, formatted as follows:

wwwwxhhhhh+xxxx+yyyy

The values for the size may be omitted, thus also the following format is legal: **+xxx+yyy**

To update the startup command, the command line in $TSCV_HOME/bin/common/start_tscv can be adapted.

If geometry parameters are not provided on the command line, TSCV will search the X resource database. It will search for the resource variable **tscv.geometry**, and the variable string should be formatted as **wwwwxhhhhh+xxxx+yyyy**

As the X resource DB is updated during startup of the window manager by the file .Xdefaults, this file should contain the line

tscv.geometry: wwwwxhhhhh+xxxx+yyyy

Copyright per DIN 34

M UCL FILE IO VIA A SPECIFIC SAS

The UCL Language as used for CGS lacks definitions of operations to read from ordinary files resp. to write to ordinary files. Such operations are also not supported by UCL System Libraries. To overcome this situation a SAS has been developed which provides operations to Automated Procedures via a UCL User Library allowing reading or writing of ASCII strings from/to ordinary text files.

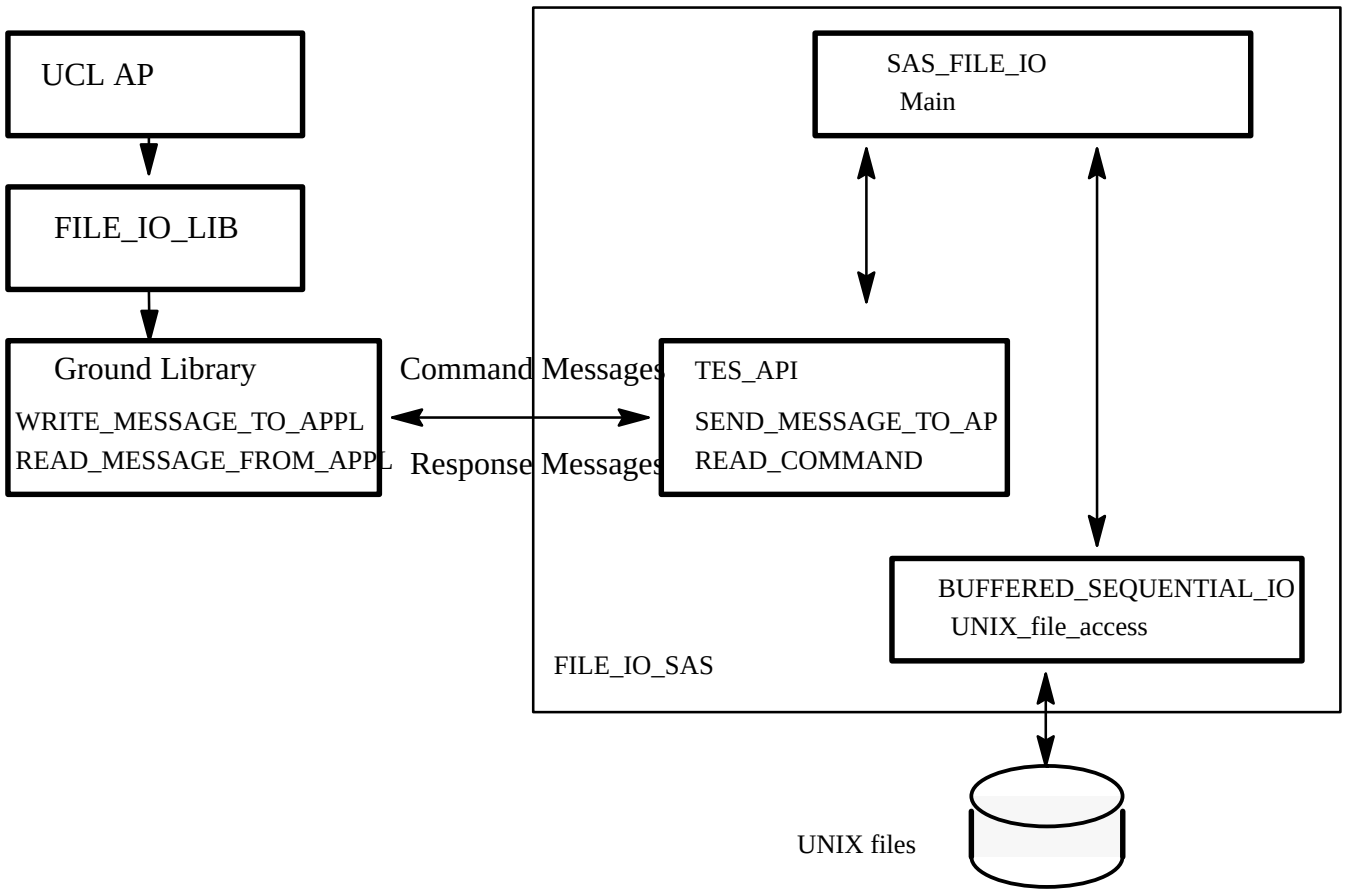
The communication between the AP and the SAS is implemented in a transparent way in the UCL User Library operations. Thus, from the user's point of view, the interface is very similar as if the operations would have been implemented in a UCL System Library.

REGISTER/UNREGISTER: Register the AP to the SAS
OPEN/CLOSE: Open a file for input or output resp close the file.
GET, GETLINE: Read text from a file
PUT,NEWLINE: Write text to a file

END_OF_FILE, END_OF_LINE, SKIP, SKIP_LINE, HANDLE_VALID: Status and control operations

To allow for deletion of files, the **DELETE** operation is available.

There is an additional operation implemented allowing execution of any UNIX command from within UCL:
EXECUTE: Execute a UNIX command via "/bin/sh"



Copyright per DIN 34

M-1 Communication between FILE_IO_LIB and SAS_FILE_IO

The procedures from FILE_IO_LIB and SAS_FILE_IO communicate by exchanging messages. Messages are simple ASCII–strings. In principle the communication between an AP which uses FILE_IO_LIB and SAS_FILE_IO is performed as follows:

- The FILE_IO_LIB procedure creates a command message which is sent to SAS_FILE_IO using the UCL function WRITE_MESSAGE_TO_APPLICATION. It then starts waiting for the response message.
- SAS_FILE_IO reads the command message and sends an acknowledge to CGS using ACKNOWLEDGE_COMMAND.
- The command message is stored in a command buffer until SAS_FILE_IO is able to process it.
- The message is parsed and the appropriate actions are executed.
- A response message containing the result of command execution is sent to the AP with the command SEND_MESSAGE_TO_AP.
- FILE_IO_LIB procedure reads the response message using READ_MESSAGE_FROM_APPLICATION and evaluates it.

For realization of this basic communication scheme the following assumptions must be met:

1. Before the AP can send a message, it has to know the Application–ID of SAS_FILE_IO. This ID is stored in the housekeeping variable SAS_ID. The full pathname of SAS_ID is specified within the body of FILE_IO_LIB.
2. SAS_FILE_IO must know the full AP–name, which is expected as a parameter value for SEND_MESSAGE_TO_AP. The AP has to inform the SAS of it's name within the first message.
3. Several instances of an AP can be connected simultaneously with SAS_FILE_IO. If more than one instance of the AP is running, a message is passed to all instances. Therefore, the AP–name is not sufficient for identification of the receiver (or sender) of a message. Every message contains an unique signature, which can be assigned to exactly one AP instance. Messages which do not contain the correct signature can safely be ignored by the AP instance. As signature first the AP–ID of the AP instance and later an unique AP–handle is used.

After determination of the Application–ID of SAS_FILE_IO the AP sends a message with it's full name to SAS_FILE_IO, which is signed with the AP–ID. SAS_FILE_IO stores the name of the AP in an internal table and assigns a unique AP–handle to the AP. The AP–handle is sent the AP in the response message, which is signed with the AP–ID. All future messages from and to this specific instance of the AP are signed with the delivered AP–handle.

During buffering and processing a command message, SAS_FILE_IO sends approximately all ?? seconds a contact message to the sender of the command. If FILE_IO_LIB does not receive either the response message or a contact message within a predefined number of seconds after sending the command message or receiving a contact message, it stops waiting and returns an connection error.

M-1.1 Messages

As already mentioned above, messages are simple ASCII strings. For separation of the various components the character ';' is used as delimiter.

Command messages

Command messages are sent from an FILE_IO_LIB procedure to SAS_FILE_IO. They can be up to 255 characters long. The format of a command message is as follows:

Copyright per DIN 34

keyword ; *AP–handle* ; *parameters*

keyword identifies the desired operation. It usually corresponds to the name of the calling FILE_IO_LIB procedure. *AP–handle* is the handle which is assigned to the AP–instance. The contents and format of *parameters* depend on the operation to be executed. Therefore, *parameters* is evaluated depending on *keyword*.

Response messages

Response messages are sent from SAS_FILE_IO to the AP. They are limited of 80 characters. The format of a response message is:

AP–handle ; *message type* ; *result*

AP–handle is the handle of the AP which sent the respective command message. *message type* is a single character:

- *message type* = ’0’ means the command executes successful (result message). In this case *result* contains the results. Format and contents of *result* depends on the performed operation.
- *message type* = ’1’ announces that an error has occurred (error message). In this case *result* contains the error message.
- *message type* = ’2’ means that SAS_FILE_IO was not yet able to process the command (contact message). In this case *result* is empty.

M-1.2 Error handling

Errors might be detected both in SAS_FILE_IO and FILE_IO_LIB. SAS_FILE_IO usually informs the user not directly about an occurred error. Instead of this, SAS_FILE_IO attempts to send the error message to the AP instance which sent the appropriate command message. Sometimes this is not possible, e.g. if the sender of the command message can not be determined. In this case the error message is passed to CGS using SEND_ERROR_MESSAGE.

If FILE_IO_LIB receives an error message or detects an error, it delivers the error message to the user using WRITE_MESSAGE_TO_USER. The procedure then returns the respective error status as result code.

Copyright per DIN 34

M-1.3 Procedures in File_IO_Lib

Each call to a procedure from FILE_IO_LIB returns a result code. Codes which can be returned by all or nearly all procedures are:

- OKAY everything o.k.
- CONNECTION_ERROR The connection to SAS_FILE_IO is broken.
- NOT_REGISTERED The AP is not connected to SAS_FILE_IO.
- INVALID_FILE_HANDLE The handle has not been associated with a file.

Some result codes should never appear under normal circumstances and indicate most likely a bug in FILE_IO_LIB or SAS_FILE_IO:

- BUFFER_OVERFLOW The command buffer in SAS_FILE_IO is full.
- MESSAGE_FORMAT_ERROR SAS_FILE_IO could not parse the command message. This should never happen. May be FILE_IO_LIB or SAS_FILE_IO is obsolete.
- DEVICE_ERROR a problem with the hardware
- END_ERROR
- DATA_ERROR
- OTHER_ERROR

M-1.3.1 REGISTERED

- **Syntax:** function REGISTERED : boolean;
- **Parameters:** none
- **Notes:** This function returns true, if the AP is connected to SAS_FILE_IO, otherwise false.

M-1.3.2 REGISTER

- **Syntax:**
```
procedure REGISTER ( in AP_NAME      : string;
                     in HOST         : NODE_NAME;
                     out FIO_RESULT  : FIO_RETURN);
```
- **Parameters:**
  - AP_NAME is the pathname of the AP.
  - HOST specifies the host, on which SAS_FILE_IO will be started if it is not already running.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the AP_NAME together with the other message components exceeds the maximal length of a command message.
ALREADY_REGISTERED	REGISTER was already called before.
REGISTER_REJECTED	There is no free AP–handle.

- **Notes:** REGISTER has to be called before using other FILE_IO_LIB procedures. It establishes the connection to SAS_FILE_IO.

Copyright per DIN 34

M-1.3.3 UNREGISTER

- **Syntax:** procedure UNREGISTER (out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - FIO_RESULT is the result code returned by this call.
- **Notes:** This procedure disconnects the AP from SAS_FILE_IO. All files, which are still opened by this AP, are closed before. SAS_FILE_IO not checks whether a registered AP still exists. It is highly recommended to call UNREGISTER before finishing the AP. Otherwise the AP–handle is permanently locked and can be released only by resetting or restarting SAS_FILE_IO.

M-1.3.4 OPEN

- **Syntax:** procedure OPEN (in FILENAME : string;
  in MODE : FILE_MODE;
  out HANDLE : FILE_HANDLE;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - FILENAME consists of the path and the name of the file to be opened.
  - MODE specifies the allowed operations for the file.

IN_FILE	opens an existing file for reading. All input (GET, GET_LINE) and status operations (END_OF_LINE, END_OF_FILE) are allowed.
OUT_FILE	creates a new file. If the file already exists, it is truncated to zero length. Only output operations (PUT, NEW_LINE, PUT_LINE) are allowed.
APPEND_FILE	opens an existing file for appending new data. Only output operations are allowed.

  - HANDLE is an unique file handle to be used in future calls to FILE_IO_LIB procedures.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the AP_NAME together with the other message components exceeds the maximal length of a command message.
TOO_MANY_OPEN_FILES	All file handles are already assigned.
FILE_LOCKED	The file is locked (see below).
NAME_ERROR	FILENAME parameter is invalid.
USE_ERROR	It is impossible to open resp. create a file with the specified name.

- **Notes:** A file can be opened simultaneously several times for reading. It is not possible to open a file, which is already opened for writing, or to open a file for writing, which is already open.

Copyright per DIN 34

M-1.3.5 CLOSE

- **Syntax:** procedure CLOSE (in HANDLE : FILE_HANDLE;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - FIO_RESULT is the result code returned by this call.
- **Notes:** none

M-1.3.6 DELETE

- **Syntax:** procedure DELETE (in FILENAME : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - FILENAME is name of the file to be deleted.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of FILENAME together with the other message components exceeds the maximal length of a command message.
FILE_LOCKED	The file is open.
NAME_ERROR	FILENAME parameter is invalid.
USE_ERROR	Deletion of the specified file is not possible.

- **Notes:** It’s not possible to delete an open file.

M-1.3.7 HANDLE_VALID

- **Syntax:** procedure HANDLE_VALID (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if HANDLE is assigned to an open file, otherwise false.
  - FIO_RESULT is the result code returned by this call.
- **Notes:** none

Copyright per DIN 34

M-1.3.8 END_OF_LINE

- **Syntax:** procedure END_OF_LINE (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if the last read character is a line feed or the last character in the file has been read.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure is only applicable for input files.

M-1.3.9 END_OF_FILE

- **Syntax:** procedure END_OF_FILE (in HANDLE : FILE_HANDLE;
  out RESULT : boolean;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - RESULT is true if the last character in the file has been read.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure is only applicable for input files.

M-1.3.10 PUT

- **Syntax:** procedure PUT (in HANDLE : FILE_HANDLE;
  in ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters which will be appended to the file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for input.
- **Notes:** This procedure is only applicable for output files. ITEM can include all kinds of characters.

Copyright per DIN 34

M-1.3.11 GET

- **Syntax:** procedure GET(in HANDLE : FILE_HANDLE;
  out ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters read from file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure reads characters from the input file into ITEM until either ITEM is full or the end of the file is reached. The characters are not interpreted anyway. In case of an attempt to read past the end of the file ITEM is empty.

M-1.3.12 SKIP

- **Syntax:** procedure SKIP(in HANDLE : FILE_HANDLE;
  in CHARS_TO_SKIP : unsigned_integer;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - CHARS_TO_SKIP is the number of characters to be ignored in the input file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure advances the input position over CHARS_TO_SKIP characters.

M-1.3.13 NEW_LINE

- **Syntax:** procedure NEW_LINE (in HANDLE : FILE_HANDLE;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for input.
- **Notes:** This procedure appends a line terminator (line feed) to the file.

Copyright per DIN 34

M-1.3.14 PUT_LINE

- **Syntax:** procedure PUT_LINE (in HANDLE : FILE_HANDLE;
  in ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters which will be appended to the file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for input.
- **Notes:** A call on PUT_LINE is equivalent to a call on PUT followed by a call on NEW_LINE.

M-1.3.15 GET_LINE

- **Syntax:** procedure GET_LINE (in HANDLE : FILE_HANDLE;
  out ITEM : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - ITEM contains the characters read from file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure fetches characters from the input file into ITEM until either ITEM is full or the end of the line is reached. The line terminator (line feed) itself not appears in ITEM. The other characters are not interpreted anyway. In case of an attempt to read past the end of the file ITEM is empty.

M-1.3.16 SKIP_LINE

- **Syntax:** procedure SKIP_LINE (in HANDLE : FILE_HANDLE;
  in LINES_TO_SKIP : unsigned_integer;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - HANDLE is the one returned from the respective OPEN.
  - LINES_TO_SKIP is the number of lines to be ignored in the input file.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

    MODE_ERROR The file was opened for output.
- **Notes:** This procedure advances to the beginning of the LINES_TO_SKIP’th following line . If there aren’t so much lines left, it advances the input position to the end of the file.

Copyright per DIN 34

M-1.3.17 EXECUTE

- **Syntax:** procedure EXECUTE (in COMMAND : string;
  out FIO_RESULT : FIO_RETURN);
- **Parameters:**
  - COMMAND is the command string to be executed.
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

COMMAND_TOO_LONG	The length of the COMMAND together with the other message components exceeds the maximal length of a command message.
PROCESS_ERROR	No new process could be created.

- **Notes:** SAS_FILE_IO creates a new process and executes COMMAND using '/bin/sh'. The created process is completely independent from SAS_FILE_IO. It is up to the user to redirect input and output if necessary. FIO_RESULT contains only information about message delivery (to SAS_FILE_IO) and process creation, but not about the created process.

M-1.3.18 ADD_FILE_TO_TEST_SESSION

- **Syntax:** procedure ADD_FILE_TO_TEST_SESSION
  (in FILE_NAME : STRING; — full UNIX pathname
  in SESSION : STRING; — name of test session
  in PRODUCER : STRING; — name identifying the producer of the file
  in CREATION_TIME: TIME ; — time when file was created
  out FIO_RESULT : FIO_RETURN);

  — Store a file in the specified Test Execution Session within the Test Result
  — Database(TRDB). File is transfered to the TRDB directory and maintained as
  — part of the test session
  — Note: To get the currently open test session, the user is obliged to read
  — the name from a SW Variable / HK value before calling this operation

- **Parameters:**
  - FILE_NAME is the name of the file to be added (full pathname)
  - SESSION is the name of the test session
  - PRODUCER is the name software item having created the file
  - CREATION_TIME: time when file was created
  - FIO_RESULT is the result code returned by this call. Possible result codes are:

SESSION_ERROR	Session is not defined
PROCESS_ERROR	could not create a new process to transfer/register the file

Copyright per DIN 34

M-2 Installation of SAS_FILE_IO and FILE_IO_LIB

The following instructions describe one way to integrate FILE_IO_LIB into an existing CGS system.

1. **Installation of SAS_FILE_IO**

   - Copy or link SAS_FILE_IO executable to directory $SAS_HOME/bin.
   - Enter the SAS name (normally SAS_FILE_IO), the port number and the host, on which SAS_FILE_IO will be executed, in the SYSTEM_TOPOLOGY_TABLE.

     Note: Verify, that on the node where the SAS_FILE_IO is assigned to, **no TES is assigned** as well. The SAS first looks on a locally assigned TES to connect to, and thus the connection to another test node (TES) having started the SAS would fail. Thus, every other SUN in the topology table is possible, but no other SUN being a test node.
   - Create an end item of type EGSE_SOFTWARE in the MDB and enter the required information(SAS_TYPE,SHORT_NAME). The short name defined for the SAS in this item has to correspond exactly to the name of the SAS executable file.

2. **Installation of FILE_IO_LIB**

   - Create an end item of type UCL_USER_LIBRARY and copy specification and body of FILE_IO_LIB into the created library.
   - Setup the library specification:
     - Check the pathname of the imported GROUND_LIBRARY.
   - Setup the library body:
     - Check the pathname of the imported GROUND_LIBRARY and SUPPORT_LIBRARY.
   - Compile and store library specification and body.

3. **Setup the test configuration**

   - The host on which SAS_FILE_IO will be executed has to participate on the test. Insert the appropriate entry in menu **Open–>EGSE Workstation Nodes** resp. in menu **Open–>EGSE Test Nodes** or **Open–>DB Server Node**
   - Ensure that the CDU, which holds the items for the software variable and the SAS, is loaded by a test node (check in menu **View–>EGSE Test Node Items**).
   - Assign the SAS to a test node in menu **View–>EGSE Test Node SASs**. Note that this entry has to correspond to entry in SYSTEM_TOPOLOGY_TABLE in the body of FILE_IO_LIB.
   - Generate a new Scoe file for the Test Configuration by selecting **Tools–>Generate Scoe Files**.

4. **Shutdown and restart CGS**,
   in order to let the modifications in SYSTEM_TOPOLOGY_TABLE and the generated Scoe files become effective.

Copyright per DIN 34

N MDB CONSISTENCY CHECKS

The MDB Consistency Checker Program can be called from the I_MDB program for the scopes CDU or CCU. It performes predefined checks on all the end items defined in that scope.

There is another program for specific end items that can be called for single end items. It performs checks on the data defined for this end items. References to other end items are checked within the current scope (CCU or CDU).

N-1 Consistency Checker – List of CGS Standard Consistency Checks

N-1.1 Mandatory Checks

CGS–ERR–0001 Mandatory aggregate is not defined

The definitions occur in the MDB data dictionary (DADI).

CGS–ERR–0002–1 Mandatory attribute <%s> is not defined

For single record aggregates only

The definitions occur in the MDB data dictionary (DADI).

CGS–ERR–0002–2 Mandatory attribute <%s> is not defined in record <%s>

For multi record aggregates only

The definitions occur in the MDB data dictionary (DADI).

N-1.2 Uniqueness Checks

CGS–ERR–0003 Attribute <%s> is not unique in scope <%s>

The definitions occur in the MDB data dictionary (DADI).

CGS–ERR–0004 Attribute is not unique in end item <%s>

The definitions occur in the MDB data dictionary (DADI).

CGS–ERR–0005 Foreign key <%s> is not unique in scope. Conflicting end items: <%s>

Foreign key aggregates must be unique in the configuration scope.

N-1.3 Referential Integrity Checks

CGS–ERR–0006 Pathname reference is out of scope <%s>

Constraint: Each pathname must refer to an item within the configuration scope.

CGS–ERR–0007 Pathname reference to wrong type <%s>

The definitions occur in the MDB data dictionary (DADI).

Copyright per DIN 34

N-1.4 Cross Reference Checks

The following checks are defined for end item types having Cross Reference Lists.

CGS–ERR–0009–1 Item is not compiled.
CGS–ERR–0009–2 Item is not complete and not up to date.
CGS–ERR–0009–3 Item is not complete.
CGS–ERR–0009–4 Item is not up to date.

Constraint: CLS dependency checks

CGS–ERR–0010 Pathname reference <%s> is out of date – Regeneration of this item required

The definitions occur in the MDB data dictionary (DADI).

The following table describes the consistency rules for end item types having Cross Reference Lists. The table consists out of three columns:

1. End Item Type:	End item type that has a Cross Refrence List Aggregate, e.g. FWDU_SYNOPTIC_DISPLAY
2. Referenced End Item Type:	Referenced end item type, e.g. a FWDU_SYNOPTIC_DISPLAY may reference an end item of type EGSE_INTEGER_MEASUREMENT
3. Depending Aggregate of referenced end item type:	In case one of the attributes of this aggreagte changes, the end item has to be regenerated, e.g. a FWDU_SYNOPTIC_DISPLAY may reference an end item of type EGSE_INTEGER_MEASUREMENT. The EGSE_INTEGER_MEASUREMENT owns the aggerate T_INTEGER_DANGER_LIMITS. If one of the attributes of the aggreagte changes, the FWDU_SYNOPTIC_DISPLAY has to be regenerated.

Note:
The table below defines the rules, in which cases a change of a referenced end item requires a regeneration of the end item that performs the reference. The table does not define a restriction for allowed end item type references, i.e. each Cross Reference List may reference any end item type. In case there is a restriction on referenced end item types, this has to be ensured by the application e.g. FWDU, GWDU etc.
The table does not contain CLS end item types. This end item types will be checked by the CLS dependency check which is integrated in the consistency checker (check CGS–ERR–0009).

N-1.5 Check Minimum Number of Records

CGS–ERR–0011 It must exist at least <%s> records in aggregate

In any aggregates must be exist at least “minimum number of records” for each item.

The definitions occur in the MDB data dictionary (DADI).

N-1.6 Double SID Check

CGs–ERR–0012 Fatal error on SID <%s>: The same SID found in end item <%s>

Constraint: In the configuration scope shall not exist different items with the same SID.

Copyright per DIN 34

N-1.7 Privileges Check

CGs–ERR–0013 Privileges <%s> are invalid.

Constraint: Each privilege must be valid.

N-1.8 Nickname Check

CGS–ERR–0014 Name <%s> is not conform to UCL syntax for names

Constraint: Nickname must be conform to UCL syntax for names.

Copyright per DIN 34

N-1.9 CGS Aggrgate and End Item Type related Special Checks

Raw Value Description

These checks are for all end item types having aggregate Raw Value Type.

CGS–ERR–0015 Invalid raw value type: <%s>

Constraint: The Raw Value Type must correspond to the Engineering Value type of the respective measurement or stimulus as specified in following Table.

End item type	Raw Value Type	Engineering Value Type
EGSE_INTEGER_MEASUREMENT	UNSIGNED_INTEGER	INTEGER
	SIGNED_INTEGER	
UNSIGNED_INTEGER_MEASUREMENT	UNSIGNED_INTEGER	UNSIGNED_INTEGER
EGSE_FLOAT_MEASUREMENT	UNSIGNED_INTEGER	FLOAT
	SIGNED_INTEGER	
	FLOAT	
DOUBLE_FLOAT_MEASUREMENT	UNSIGNED_INTEGER	DOUBLE_FLOAT
	SIGNED_INTEGER	
	DOUBLE_FLOAT	
EGSE_DISCRETE_MEASUREMENT	UNSIGNED_INTEGER	STATE_CODE
EGSE_BYTE_STREAM_MEASUREMENT	BYTESTREAM	STRING
EGSE_ANALOG_STIMULUS	FLOAT	FLOAT
EGSE_DISCRETE_STIMULUS	UNSIGNED_INTEGER	STATE_CODE
INTEGER_STIMULUS	UNSIGNED_INTEGER	INTEGER
	SIGNED_INTEGER	
BOOLEAN_STIMULUS	SIGNED_INTEGER	BOOLEAN
PULSE_STIMULUS	SIGNED_INTEGER	PULSE
BURST_PULSE_STIMULUS	UNSIGNED_INTEGER	BURST_PULSE
DOUBLE_FLOAT_STIMULUS	UNSIGNED_INTEGER	DOUBLE_FLOAT
	SIGNED_INTEGER	
UNSIGNED_INTEGER_STIMULUS	UNSIGNED_INTEGER	UNSIGNED_INTEGER

CGS–ERR–0016 Value <%s> out of range <%s>..<%s>

Check of aggregate Raw Value Size in Bits.

Constraint: The Raw Value Size is dependent on the Raw Value Type of the respective measurement or stimulus as specified in following Table.

Raw Value Type	Raw Value Size in Bits
UNSIGNED_INTEGER	0 .. 32
SIGNED_INTEGER	0 .. 32
FLOAT	32
DOUBLE_FLOAT	64

Copyright per DIN 34

CGS–ERR–0017–1 Low Value <%s> is not less than high value <%s>

For measurements only

Check of the following aggregates:
Unsigned Integer Raw Value Range
Integer Raw Value Range
Float Raw Value Range
Double Float Raw Value Range

Constraint: Low Value < High Value

CGS–ERR–0017–2 Low Value <%s> is not less or equal than high value <%s>

For other item types as measurements only

Check of the following aggregates:
Unsigned Integer Raw Value Range
Integer Raw Value Range
Float Raw Value Range
Double Float Raw Value Range

Constraint: Low Value <= High Value

Engineering Value Description

These checks are for all end item types having one of the following aggregates:
Integer Engineering Range
Unsigned Integer Engineering Range
Float Engineering Range
Double Float Engineering Range

CGS–ERR–0018–1 Low Value <%s> is not less than high value <%s>

For measurements only

Constraint: Low Value < High Value

CGS–ERR–0018–2 Low Value <%s> is not less or equal than high value <%s>

For other item types as measurements only

Constraint: Low Value <= High Value

Analog Calibration and Decalibration

These checks are for all end item types having aggregate Curve Type.

CGS–ERR–0019 Row <%s>: Value <%s> out of Raw Value Range <%s>...<%s> for the corresponding end item

Check of Analog Calibration Point Pairs.Raw Value or
Check of Analog Decalibration Point Pairs.Raw Value

Constraint: The value of this attribute must be within the limits defined by the raw value range aggregate of the corresponding end item.

Copyright per DIN 34

CGS–ERR–0020 Row <%s>: Value <%s> out of Engineering Value Range <%s>...<%s> for the corresponding end item

Check of Analog Calibration Point Pairs.Engineering Value or
Check of Analog Decalibration Point Pairs.Engineering Value

Constraint: The value of this attribute must be within the limits defined by the engineering value range aggregate of the corresponding end item.

CGS–ERR–0021 Point pairs describe not a monotonous function

Check of Analog Calibration Point Pairs or
Check of Analog Decalibration Point Pairs

Constraint: the sequence of N Raw Values must be monotonic over the specified raw value range
i.e. $RAW\_VALUE_i < RAW\_VALUE_{i+1}$
or $RAW\_VALUE_i > RAW\_VALUE_{i+1}$ for i=1 .. N–1

Constraint: the sequence of N Eng. Values must be monotonic over the specified eng. value range
i.e. $ENG\_VALUE_i <= ENG\_VALUE_{i+1}$
or $ENG\_VALUE_i >= ENG\_VALUE_{i+1}$ for i=1 .. N–1

Discrete Calibration

These checks are for all end item types having aggregate Discrete Calibration.

to be continued in NO TAG, NO TAG and NO TAG

CGS–ERR–0022 number of calibration records <%s> is not less or equal 2**Raw Value Size in Bits <%s>

Check of Discrete Calibration

Constraint: The actual number of records per end item in the Discrete Calibration / Decalibration aggregate must be <= `2**Raw_Value_Size_in_Bits`, with `Raw_Value_Size_in_Bits <= 32`

CGS–ERR–0023 Undefined Values State Code <%s> is identical to calibration entry

Check of Undefined Values State Code

Constraint: For any given *discrete* end–item (measurement, stimulus, SW variable) the Undefined Values State Code, if specified, must differ from the state codes defined in the T_DISCRETE_CALIBRATION aggregate.

Byte Stream Calibration

These checks are for all end item types having aggregate Byte Stream Calibration.

CGS–ERR–0024 Value <%s> exceeds Raw Value Size in Bytes <%s>

Check of Byte Stream Calibration.Position

Constraint: Position must be <= Raw Value Size in Bytes of given end item

Copyright per DIN 34

CGs–ERR–0025 Position <%s> + Length <%s> exceeds Raw Value Size in Bytes <%s> + 1

Check of Byte Stream Calibration.Length

Constraint: Position + Length <= Raw Value Size in Bytes + 1

Initial Values

These checks are for all end item types having a Initial Value aggregate.

CGS–ERR–0026 Value <%s> out of Integer Engineering Range <%s>...<%s>

Check of Initial Integer Value

Constraint: Value must be within the range defined by the Integer Engineering Range aggregate for the particular SW Variable.

CGS–ERR–0027 Value <%s> out of Float Engineering Range <%s>...<%s>

Check of Initial Float Value

Constraint: Value must be within the range defined by the Float Engineering Range aggregate for the particular SW Variable.

CGS–ERR–0028 Value <%s> out of Unsigned Integer Engineering Range <%s>...<%s>

Check of Initial Unsigned Integer Value

Constraint: Value must be within the range defined by the Unsigned Integer Engineering Range aggregate for the particular SW Variable.

CGS–ERR–0029 Value <%s> out of Double Float Engineering Range <%s>...<%s>

Check of Initial Double Float Value

Constraint: Value must be within the range defined by the Double Float Engineering Range aggregate for the particular SW Variable.

Monitoring

Copyright per DIN 34

CGS–ERR–0030–1 Referenced end item <%s> has parameters that are not optional

Check of (xxx: Integer, Float, Unsigned Integer or Double Float):
xxx Danger Limits.High Message
xxx Danger Limits.High Action
xxx Danger Limits.Low Message
xxx Danger Limits.Low Action
xxx Danger Limits.Delta Message
xxx Danger Limits.Delta Action
xxx Nominal Limits.High Message
xxx Nominal Limits.High Action
xxx Nominal Limits.Low Message
xxx Nominal Limits.Low Action
xxx Nominal Limits.Delta Message
xxx Nominal Limits.Delta Action
Discrete Monitor List.Exception Message (Expected Value.Exception Message)
Discrete Monitor List.Exception Action (Expected Value.Exception Action)
Byte Stream Monitor List.Exception Message
Byte Stream Monitor List.Exception Action
Boolean Monitoring.Exception Message
Boolean Monitoring.Exception Action
Parameter Value Change.Value Change Action

Constraint: If defined, the parameters of the referenced end item must all be optional with assigned default values.

CGS–ERR–0030–2 Referenced AP <%s> has parameters

Check of

(xxx: Integer or Float):
xxx Danger Limits.High Action
xxx Danger Limits.Low Action
xxx Danger Limits.Delta Action
xxx Nominal Limits.High Action
xxx Nominal Limits.Low Action
xxx Nominal Limits.Delta Action
Discrete Monitor List.Exception Action (Expected Value.Exception Action)
Byte Stream Monitor List.Exception Action
Parameter Value Change.Value Change Action

Constraint: If the referenced end item of type UCL_AUTOMATED_PROCEDURE than is it not allowed that the AP has parameters.

CGS–ERR–0031 Length Size in Bits must be greater than 0 or End Sign must be defined

Check of Dynamic String Parameter

Constraint: Length Size in Bits must be greater than 0 or End Sign must be defined

CGS–ERR–0032 Row <%s>: Value out of range of Danger Limits

Check of:
Integer Nominal Limits

Copyright per DIN 34

Float Nominal Limits
Unsigned Integer Nominal Limits
Double Float Nominal Limits

Constraint: Nominal High Limit <= Danger High Limit
Nominal Low Limit => Danger Low Limit
Nominal_Delta_Limit < Danger_Delta_Limit

CGS–ERR–0033 High Limit <%s> is not greater than Low Limit <%s>

Check of:
Integer Danger Limits
Float Danger Limits
Unsigned Integer Danger Nominal Limits
Double Float Danger Limits

Constraint: Danger High Limit > Danger Low Limit

CGS–ERR–0034 Row <%s>: High Limit is not greater than Low Limit

Check of:
Integer Nominal Limits
Float Nominal Limits
Unsigned Integer Nominal Nominal Limits
Double Float Nominal Limits

Constraint: Nominal High Limit > Nominal Low Limit

CGS–ERR–0035 Expected Value <%s> must be defined in Discrete Calibration

Check of Discrete Monitor List (Expected Value)

Constraint: If monitoring is specified in Discrete Monitor List, the expected values must be defined in Discrete Calibration.

CGS–ERR–0036 Out of Integer Engineering Range

Check of:
Integer Danger Limits.Danger High Limit
Integer Danger Limits.Danger Low Limit

Constraint: Danger High Limit <= High Value in engineering range
Danger Low Limit => Low Value in engineering range

CGS–ERR–0037 Out of Float Engineering Range

Check of:
Float Danger Limits.Danger High Limit
Float Danger Limits.Danger Low Limit

Constraint: Danger High Limit <= High Value in engineering range
Danger Low Limit => Low Value in engineering range

Copyright per DIN 34

CGS–ERR–0038 Out of Unsigned Integer Engineering Range

Check of:
Unsigned Integer Danger Limits.Danger High Limit
Unsigned Integer Danger Limits.Danger Low Limit

Constraint: Danger High Limit <= High Value in engineering range
Danger Low Limit => Low Value in engineering range

CGS–ERR–0039 Out of Double Float Engineering Range

Check of:
Double Float Danger Limits.Danger High Limit
Double Float Danger Limits.Danger Low Limit

Constraint: Danger High Limit <= High Value in engineering range
Danger Low Limit => Low Value in engineering range

Conditions

These Checks are defined for all end item types having a condition aggregate.

CGS–ERR–0040–1 Value 1 <%s>, Value 2 <%s>: Value 2 must be greater or equal than Value 1

Check of Integer Conditions.Value 2, Float Conditions.Value 2 or Unsigned Integer Conditions.Value 2

Constraint: If given, Value_2_(High_value) >= Value_1_(Low_Value)

CGS–ERR–0040–2 Row <%s>, Operator <%s>, Value 2 <%s>: in case Value 2 is defined, Operator has to be ”in_range

Check of Integer Conditions.Value 2, Float Conditions.Value 2 or Unsigned Integer Conditions.Value 2

Constraint: If given, Operator must be ”in_range”

CGS–ERR–0040–3 Row <%s>: in case Operator is ”in_range”, Value 2 has to be defined

Check of Integer Conditions.Value 2, Float Conditions.Value 2 or Unsigned Integer Conditions.Value 2

Constraint: Must be given, if operator is ”in_range”

CGS–ERR–0043–1 Row <%s>, Action Type ”START_AP”: referenced end ’ || ’item has to be of type UCL_AUTOMATED_PROCEDURE

Check of Conditions.Action Enditem Reference

Constraint: If Action_Type is ”START_AP”, the pathname must be of type UCL_AUTOMATED_PROCEDURE.

CGS–ERR–0043–2 Row <%s>, Action Type <%s>: referenced end ’ || ’item must not be of type UCL_AUTOMATED_PROCEDURE

Check of Conditions.Action Enditem Reference

Constraint: If Action_Type is not ”START_AP”, the pathname must not be of type UCL_AUTOMATED_PROCEDURE.

Copyright per DIN 34

CGS–ERR–0045–1 Row <%s>: in case Action Type is "SWITCH_LIMIT_SET", the referenced end item <%s> has to have nominal limit sets defined

Check of Conditions.Action Enditem Reference

Constraint: If Action_Type = 'Switch_Limit_Set', the referenced enditem must have nominal limit sets defined.

CGS–ERR–0045–2 Row <%s>: in case Action Type is "SWITCH_LIMIT_SET", the given Limit Set Number <%s> has to be defined for <%s>

Check of Conditions.Limit Set Number

Constraint: If Action_Type = 'Switch_Limit_Set', the given Limit Set Number must be defined for the Action_Enditem_Reference.

CGS–ERR–0047 Row <%s>: referenced AP <%s> has parameters

Check of:
Integer Conditions.Action Enditem Reference
Unsigned Integer Conditions.Action Enditem Reference
Float Conditions.Action Enditem Reference
Discrete Conditions.Action Enditem Reference
Byte Stream Conditions.Action Enditem Reference

Constraint: Referenced UCL_AUTOMATED_PROCEDUREs must not have parameters defined.

CGS–ERR–0048 Value <%s> must be defined in Discrete Calibration.

Check of Discrete Conditions.Value

Constraint: If Discrete Conditions are specified, the values compared must be defined in Discrete Calibration.

CGS–ERR–0049 Mandatory aggregate is not defined

Check the existence of aggregate T_CURVE_TYPE for the following end item types:
EGSE_INTEGER_MEASUREMENT
EGSE_FLOAT_MEASUREMENT
DOUBLE_FLOAT_MEASUREMENT

Constraint: The curve type is mandatory if raw value type is not equal than engineering value type.

EGSE Physical Address

CGS–ERR–0053 Value <%s> out of range <%s>..<%s>

Check of EGSE Physical Address

Constraint: The allowed range is 0 .. MAX_SLOT_CHANNEL, where MAX_SLOT_CHANNEL is an integer number dependent on the Slot Class as per table below.
Note: Check shall be performed only when remote terminal Slot Class is not UNDEFINED or NON_STANDARD.

Copyright per DIN 34

Slot Class	BITS_1	BITS_8	BITS_16	BITS_32	PACKET
MAX_SLOT_CHANNEL	511	63	31	15	0

EGSE_NODE

Check of end item type EGSE_NODE

CGS–ERR–0054 CGS Prefix <%s> has to be <%s> in case Node Type is <%s>

Check of CGS Internal Name.CGS Prefix

Constraint: CGS Prefix must be:
”TES” if Node Type is TEST_NODE
”CSS” if Node Type is SIMULATOR
”HCI” if Node Type is WORKSTATION
”DBS” if Node Type is DATABASE_SERVER
“CIS” if Node Type is CIS

CGS–ERR–0055 Combination of Node Type <%s> and CGS Internal Number <%s> is not unique. Conflicting end items: <%s>

Check of CGS Internal Name.CGS Internal Number

Constraint: CGS internal names must be unique, i.e. no two end items of type EGSE NODE may have the same CGS Prefix and Internal Number.

CGS–ERR–0056 Has to be ”1” (is <%s>) for nodes of type <%s>

Check of CGS Internal Name.CGS Internal Number

Constraint: CGS Internal Number must be 1 if CGS Prefix is ”DBS”.

CGS–ERR–0057 Blanks or a blank name are not allowed

Check of Logical Name.Logical Name

Constraint: Blanks or a blank name shall not be allowed

EGSE_SOFTWARE

Check of end item type EGSE_SOFTWARE

CGS–ERR–0058 Blanks or a blank name are not allowed

Check of Short Name.Short Name

Constraint: The short name may not contain any space (blank) characters.
It may not be the empty string.

EGSE_TEST_CONFIGURATION

Check of end item type EGSE_TEST_CONFIGURATION

CGS–ERR–0059 Referenced end item does not describe a database server: <%s>

Check of EGSE Database Node.Database Node

Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = DATABASE_SERVER.

Copyright per DIN 34

CGS–ERR–0060 Referenced end item does not describe a simulator: <%s>

Check of Simulator Nodes.Simulator Node

Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = SIMULATOR.

CGS–ERR–0061 Referenced end item does not describe a workstation or a CGS interface server (CIS): <%s>

Check of EGSE Workstation Nodes.Workstation Node

Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = WORKSTATION or Node Type = CIS.

CGS–ERR–0062–1 No workstation present for test configuration

Check of EGSE Workstation Nodes.Workstation Node

Constraint: At least one workstation has to be present per test configuration.

CGS–ERR–0062–2 No workstation participating in test configuration

Check of EGSE Workstation Nodes.Is Participating

Constraint: There must be at least one workstation participating in a given test configuration.

CGS–ERR–0064 Referenced end item does not describe a test node: <%s>

Check of EGSE Test Nodes.Test Node

Constraint: The specified pathname must refer to an existing end item of type EGSE Node with Node Type = TEST_NODE.

CGS–ERR–0065–1 No testnode present for test configuration

Check of EGSE Test Nodes.Test Node

Constraint: There must be at least one Test Node per test configuration.

CGS–ERR–0065–2 Not exactly one test node is Master Test Processor per test configuration

Check of EGSE Test Nodes.Is Master Test Processor

Constraint: There must be one (and only one) Master Test Processor per test configuration.

CGS–ERR–0065–3 No test node participating in test configuration

Check of EGSE Test Nodes.Is Paticipating

Constraint: There must be at least one test node participating in a given test configuration.

CGS–ERR–0066 test node <%s>: loaded ADU description <%s> has out of scope references: <%s>

Check of relationships between EGSE Test Node Items, MEASUREMENT_LIST_TABLE, PUS_STRUCTURE_ID and ADU Description end items

Constraint: Enditems of type MEASUREMENT_LIST_TABLE and PUS_STRUCTURE_ID referenced in enditems of type xxx_ADU_DESCRIPTIONs (directly or indirectly via MEASUREMENT_LIST_TABLE enditems) shall exist in the same scope of specified items for one test node.

Copyright per DIN 34

CGS–ERR–0068 Test configuration contains a combination of REPLAY and SIMULATION nodes

Check of EGSE Test Nodes.Execution Mode

Constraint: A test configuration must never contain nodes executing in a combination of REPLAY and SIMULATION mode.

CGS–ERR–0069 End item must actually have a parameter list with all parameters being optional, i.e. having default values: <%s>

Check of EGSE Test Nodes.Initial Automated Procedure

Constraint: If defined, the parameters of the referenced AP must all be optional with assigned default values.

CGS–ERR–0070 No correspondence to aggregate EGSE Test Nodes: <%s>

Check of EGSE Test Node Items.Test Node

Constraint: This is a reference to one of the Test Nodes specified in the EGSE Test Nodes aggregate.

CGS–ERR–0071–1 Item <%s> is loaded into more than one test node <%s>
CGS–ERR–0071–2 Subtree <%s> and contained items <%s> are loaded into more than one test node <%s>

Check of EGSE Test Node Items.Loaded Item

Constraint: If the given item appears in more than one Loaded Item List (i.e. if the items are loaded into more than one test node), then this item must not be of type:

- EGSE_xxx_SW_VARIABLE
- EGSE_xxx_DERIVED_VALUE
- PUS_ADU_DESCRIPTION
- UNSTRUCTURED_ADU_DESCRIPTION
- STRUCTURED_ADU_DESCRIPTION
- CCSDS_ADU_DESCRIPTION
- EGSE_ANALOG_STIMULUS
- EGSE_DISCRETE_STIMULUS
- EGSE_PREDEFINED_TC
- PUS_TC
- EGSE_BINARY_PACKET
- EGSE_MONITOR_LIST
- GDU_DESCRIPTION_LIST

CGS–ERR–0072 No item is downloaded into testnode <%s>

Check of EGSE Test Node Items.Loaded Item

Constraint: At least one item must be downloaded into each test node.

Copyright per DIN 34

CGS–ERR–0073 test node <%s>: loaded GDU_DESCRIPTION_LIST <%s> has out of scope references: <%s>

Check of relationships between EGSE Test Node Items, GDU items and GDU Description List items

Constraint: The list of items loaded for one test node must contain all end items of type EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, PUS_TC or EGSE_BINARY_PACKET which are referenced in all end items of type GDU_DESCRIPTION_LIST in this list of items.

CGS–ERR–0075 test node <%s>: loaded SIMULATED_ADU_DESCRIPTION <%s> has out of scope references: <%s>

Check of relationships between EGSE Test Node Items, ADU_DESCRIPTION and SIMULATED_ADU_DESCRIPTION end items

Constraint: The xxx_ADU_DESCRIPTIONs referenced in all SIMULATED_ADU_DESCRIPTION end items contained in all the specified items for one test node shall exist within the same scope.

CGS–ERR–0076 test node <%s>: loaded EGSE_MONITOR_LIST <%s> has out of scope references: <%s>

Check of relationships between EGSE Test Node Items, EGSE_SW_xxx_VARIABLE, ESGE_xxx_DERIVED_VALUE and EGSE_MONITOR_LIST end items

Constraint: The list of items loaded for one test node must contain all end items of type EGSE_SW_xxx_VARIABLE or ESGE_xxx_DERIVED_VALUE which are referenced in all end items of type MONITOR_LIST and in this list of items.

CGS–WRN–0077 test node <%s>: Measurement <%s> is not referenced in an ADU description in the test node scope

Check of relationships between EGSE Test Node Items, EGSE Measurement, EGSE_MONITOR_LIST and ADU Description end items

Constraint: The EGSE_xxx_MEASUREMENTs referenced in all MONITOR_LIST or PUS_STRUCTURE_ID end items contained in all the specified items for one test node shall also be referenced in xxx_ADU_DESCRIPTION end items contained in the same scope (either via direct references or via indirekt references on STRUCTURE_IDs or via indirect references on MEASUREMENT_LIST_TABLEs and STRUCTURE_IDs).

CGS–ERR–0078 measurement <%s> referenced on different test nodes <%s> by ADUs <%s>

Check of relationships between EGSE Test Node Items, ADU Description and EGSE Measurement end items.

Constraint: All xxx_ADU_DESCRIPTIONs which have references to the same EGSE_xxx_MEASUREMENT end item (either via direct references or via indirect references on STRUCTURE_IDs or via indirect references on MEASUREMENT_LIST_TABLEs and STRUCTURE_IDs) shall exist in the same scope of specified items for one test node.

Copyright per DIN 34

CGS–WRN–0079 test node <%s>: loaded UCL_AUTOMATED_PROCEDURE <%s> has out of test node references: <%s>

Check of relationships between EGSE Test Node Items, end items referenced in UCL_AUTOMATED_PROCEDURE and UCL_AUTOMATED_PROCEDURE end items

Constraint: The UCL_USER_LIBRARY, EGSE_MONITOR_LIST, GDU_DESCRIPTION_LIST, EGSE_NODE, EGSE_SOFTWARE, EGSE_USER_MESSAGE, xxx_ADU_DESCRIPTION and VIRTUAL end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for one test node shall exist within the same scope.

CGS–ERR–0080 test node <%s>: loaded UCL_AUTOMATED_PROCEDURE <%s> has out of scope (whole test configuration) references: <%s>

Check of relationships between EGSE Test Node Items, end items referenced in UCL_AUTOMATED_PROCEDURE and UCL_AUTOMATED_PROCEDURE end items

Constraint: The EGSE_xxx_SW_VARIABLE, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_STIMULUS, EGSE_BINARY_PACKET, PUS_TC and EGSE_PREDEFINED_TC end items referenced in all UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration shall exist within the same scope.

CGS–ERR–0081 Measurement <%s> is not referenced in an ADU description for the whole test configuration

Check of relationships between EGSE Test Node Items, EGSE Measurement, EGSE SW Variable, EGSE Derived Value, GDU and ADU Description end items

Constraint: The EGSE_xxx_MEASUREMENT end items referenced in enditems of type UCL_AUTOMATED_PROCEDURE, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_MEASUREMENT, EGSE_xxx_SW_VARIABLE, EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, PUS_TC and EGSE_BINARY_PACKET contained in all the specified items for a whole test configuration shall be referenced at least in one xxx_ADU_DESCRIPTION of the same scope (either via direct references or via indirect references on STRUCTURE_IDs or via indirect references on MEASUREMENT_LIST_TABLEs and STRUCTURE_IDs)

CGS–ERR–0082 UCL_AUTOMATED_PROCEDURE <%s> references SWOP_COMMAND <%s> which does not have the mandatory aggregate "EGSE Stimulus General Info" defined

Check of relationships between EGSE Test Node Items, SWOP_COMMAND and UCL_AUTOMATED_PROCEDURE end items

Constraint: For SWOP_COMMANDS referenced in UCL_AUTOMATED_PROCEDURE end items contained in all the specified items for a whole test configuration the aggregate T_STIMULUS_GENERAL_INFO is mandatory.

Copyright per DIN 34

CGS–ERR–0083 End item <%s> has out of scope (whole test configuration) references: <%s>

Check of relationships between EGSE Test Node Items, GDU, EGSE Derived Value, EGSE Measurement, EGSE SW Variable end items and end items which are referenced in such items

Constraint: The end items referenced in EGSE_xxx_STIMULUS, EGSE_PREDEFINED_TC, PUS_TC, EGSE_BINARY_PACKET, EGSE_xxx_DERIVED_VALUE, EGSE_xxx_MEASUREMENT and EGSE_xxx_SW_VARIABLE enditems contained in all the specified items for a whole test configuration shall exist within the same scope.

CGS–ERR–0085 No correspondence to aggregate EGSE Test Nodes: <%s>

Check of EGSE Test Node SASs.Test Node

Constraint: This is a reference to one of the Test Nodes specified in the EGSE Test Nodes aggregate.

CGS–ERR–0087 test node <%s>: Attribute is not unique: Used SAS <%s>

Check of EGSE Test Node SASs.Used SAS

Constraint: A SAS shall only appear once in the list of SASes, i.e. every SAS is unique within one list.

CGS–ERR–0088 test node <%s>: Attribute is not unique: Private ID <%s> (end items: <%s>)

Check of EGSE Stimulus General Info.Private Identifier

Constraint: The Private ID string must be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

Check of ADU General Info.Private ID

Constraint: The string has to be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

Check of Response Packet.Private ID

Constraint: The Private ID string must be unique within the list of CDUs loaded into a test node for every test node and every test configuration in a CCU.

EGSE_USER_MESSAGE

Check of end item type EGSE_USER_MESSAGE

CGS–ERR–0089Number of parameters contained in the long text <%s> does not correspond to number of parameters in the Formal Parameter List <%s>

Check of Message Text.Long Text

Constraint: The number of parameters contained in the Long Text must correspond to the formal parameter list defined for the end item.

CGS–ERR–0090 Name of parameter contained in the long text at position <%s> does not correspond to Formal Parameter List for this end item: <%s>

Check of Message Text.Long Text

Constraint: The names (Pi) of the parameters contained in the Long Text must correspond to the formal Name of the parameters defined for the end item.

Copyright per DIN 34

CCSDS Packet Header

CGS–ERR–0091 Packet Length is out of range 9 .. 4095: <%s>

Check of CCSDS Primary Header.Packet Length

Constraint: If the CCSDS Packet has a secondary header, then the range of the Packet Length Field shall be 9 .. 4095 (since the secondary header is 10 bytes long), i.e.
if T_CCSDS_HEADER_DESCRIPTION.**Secondary Header** = 'TRUE' then
9 <= T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** <= 4095

Stimulus Definition

Checks for GDUs

LOCATION :	T_GENERAL_BITSTREAM_LAYOUT. **Location**	
VALUE_SIZE:	T_INTEGER_DEFINITION.**Number of Bits**,	for integer
	32,	for float
	(size_of T_BINARY_DEFINITION.**Value**) * 8 ,	for strings
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**	
Buffer Length or Global Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**	
N:	Number of entries (predefined values) in the data buffer	
PARAM_LOC :	T_LIST_OF_PARAMETERS.**Parameter Location**	
PARAM_SIZE:	T_LIST_OF_PARAMETERS.**Parameter Number of Bits**	

CGS–ERR–0092–1 Position <%s>: Location <%s> + Value Size <%s> – 1 is not less or equal than (Packet Length Field + 1) * 8 <%s>

Check of EGSE_PREDEFINED_TC.General Bitstream Layout.Location or
Check of PUS_TC.General Bitstream Layout.Location

Constraint: Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.
LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8

CGS–ERR–0092–2 Position <%s>: Location <%s> + Value Size <%s> – 1 is not less or equal than Global Length * 8 <%s>

Check of EGSE_BINARY_PACKET.General Bitstream Layout.Location

Constraint: Each Predefined Value shall be entirely located within the bounds of the data buffer, i.e.
LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

Copyright per DIN 34

CGS–ERR–0093–1 ***Parameter <%s>:" Packet Part" = DATA and "CCSDS Primary Header.Secondary Header" = TRUE: "Location" <%s> + "Number of Bits" <%s> – 1 > ("CCSDS Primary Header.Packet Length" <%s> + 5) * 8***

CGS–ERR–0093–2 ***Parameter <%s>: "Packet Part" = DATA and "CCSDS Primary Header.Secondary Header" = FALSE: "Location" <%s> + "Number of Bits" <%s> – 1 > ("CCSDS Primary Header.Packet Length" <%s> + 1) * 8***

CGS–ERR–0093–3 ***Parameter <%s>: "Packet Part" = HEADER: "Location" <%s> + "Number of Bits" <%s> – 1 > 6 * 8***

CGS–ERR–0093–4 ***Parameter <%s>: "Packet Part" = SECONDARY_HEADER: "Location" <%s> + "Number of Bits" <%s> – 1 > 4 * 8***

Check uf PUS_TC.T_LIST_OF_PARAMETERS

Constraint: Each Parameter Value shall be entirely located within the bounds of the selected part of the packet, i.e.
if PACKET_PART = DATA and SECONDARY_HEADER = TRUE then
PARAM_LOC + PARAM_SIZE – 1 <= (Packet Length Field + 5)
if PACKET_PART = DATA and SECONDARY_HEADER = FALSE then
PARAM_LOC + PARAM_SIZE – 1 <= (Packet Length Field + 1) * 8
if PACKET_PART = HEADER then
PARAM_LOC + PARAM_SIZE – 1 <= 6 * 8
if PACKET_PART = SECONDARY_HEADER then
PARAM_LOC + PARAM_SIZE – 1 <= 4 * 8

CGS–ERR–0094–1 ***Overlapping occurs: Postition <%s> is overlapped by <%s>***

CGS–ERR–0094–2 ***No record found for Position <%s> in corresponding aggregate <%s>***

Check of General Bitstream Layout.Location

Constraint: Predefined Values shall not overlap one another, i.e.:
For any Predefined Value i (i = 1 .. N–1)
$LOCATION_{(i+1)} > LOCATION_{(i)} + VALUE\_SIZE_{(i)} - 1$

CGS–ERR–0095 ***Position <%s>, Value <%s>: not in range defined by number of bits <%s>***

Check of Integer Definition.Integer Value and Unsigned Integer Definition.Unsigned Integer Value

Constraint: The value range is limited by the number of bits defined for this entity.

CGS–ERR–0096 ***no entry in the Parameter Insertion List: <%s>***

Check of List of Parameters.Parameter Name

Constraint: For each formal parameter (in aggregate T_FORMAL_PARAMETERS), there must be an entry in the Parameter Insertion List (aggregate T_LIST_OF_PARAMETERS).

Copyright per DIN 34

CGS–ERR–0097–1 number of bits does not comply with type of parameter <%s>
CGS–ERR–0097–2 number of bits is not lower or equal than raw value size of parameter <%s>

Check of List of Parameters.Number of Bits

Constraint: If Parameter Raw value Type is definded (which is the case when the Parameter value is to decalibrated), then:
Parameter Number Of Bits shall be <= Max. No. Of Bits
with Max. No. Of Bits being either
(a) the user–specified Parameter Raw value Size in Bits
(or Parameter Raw value Size in Bytes * 8 for the byte stream type)
or
(b) the default type–specified size, i.e. 32 for integer/float, 64 for double float, 255*8 for byte stream
Otherwise (i.e. Parameter Raw Value Type is not defined), the following constraints shall apply:

for a Param of SW Type:	**the No. of Bits must be:**
STRING_TYPE	<= 255*8
STATE_CODE_TYPE	<= 64
INTEGER_TYPE	<= 32
REAL_TYPE	= 32
LONG_REAL_TYPE	= 64
PATHNAME_TYPE	= 32 (SID)
TIME_TYPE	<= 40
UNSIGNED_INTEGER_TYPE	<= 32
BYTE_TYPE	<= 8
WORD_TYPE	<= 16
LONG_WORD_TYPE	<= 32
BYTE_STRING_TYPE	<= 255*8

CGS–ERR–0098–1 Parameter <%s>: Location <%s> + Value Size <%s> – 1 is not less or equal than (Packet Length Field + 1) * 8 <%s>

Check of EGSE_PREDEFINED_TC.List of Parameters.Location

Constraint: Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.
PARAM_LOC + PARAM_SIZE – 1 <= (Packet Length Field + 1) * 8

CGS–ERR–0098–2 Parameter <%s>: Location <%s> + Value Size <%s> – 1 is not less or equal than Global Length * 8 <%s>

Check of EGSE_BINARY_PACKET.List of Parameters.Location

Constraint: Each Parameter Value shall be entirely located within the bounds of the data buffer, i.e.
PARAM_LOC + PARAM_SIZE – 1 <= (Buffer Length) * 8

Copyright per DIN 34

CGS–ERR–0099–1 The Parameter Alignment attribute shall be specified only if the Location Specification Mode is ALIGNED.

Check of Parameter Position Rule.Parameter Alignment

Constraint: The Parameter Alignment attribute shall be specified if and only if the Location Specification Mode is 'ALIGNED'.

CGS–ERR–0099–2 The Size of String Length Field attribute shall be specified only if the Location Specification Mode is ALIGNED or RELATIVE.

Check of Parameter Position Rule.Size of String Length Field

Constraint: The Size of String Length Field attribute shall be specified if and only if the Location Specification Mode is 'ALIGNED' or 'RELATIVE'.

Command Verification

Check of aggregates Command Verification and Command Verification Times in GDUs

CGS–ERR–0100 Value <%s> is out of range 0.0 .. 86400.0

Check of Command Verification Times.Activation Delay in Seconds and Verification Times.Verification Timeout in Seconds

Constraint: 0.0 <= Value <= 86400.0

CGS–ERR–0102–1 Measurement to be checked <%s> is of type <%s>, Operator has to be either = or /=

Check of Command Verification.Operator

Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_... , EGSE_BYTE_STREAM_... or EGSE_STRING_..., Operator must be one of = or /=.

CGS–ERR–0102–2 Measurement to be checked <%s>: Value <%s> has to be convertible to a float value

Check of Command Verification.Value and Command Verification.High Value for range

Constraint: If Measurement_to_be_checked is of type EGSE_FLOAT_... , strings given in Value and High_Value_for_Range must be convertible to a float value.

CGS–ERR–0102–3 Measurement to be checked <%s>: Value <%s> has to be convertible to an integer value

Check of Command Verification.Value and Command Verification.High Value for range

Constraint: If Measurement_to_be_checked is of type EGSE_INTEGER_..., strings given in Value and High_Value_for_Range must be convertible to an integer value.

CGS–ERR–0102–4 Measurement to be checked <%s>: Value <%s> must be limited to 8 characters (state code)

Check of Command Verification.Value

Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_... , string given in Value must be limited to 8 characters (state code)

Copyright per DIN 34

CGS–ERR–0102–5 Measurement to be checked <%s>: Operator is ”in_range”, High Value for Range must be defined

Check of Command Verification.High Value for range

Constraint: If Operator is ”in_range”, both Value and High_Value_for_Range must ge given.

CGS–ERR–0102–6 Measurement to be checked <%s>: High Value for Range <%s> must be equal or greater then Low Value for Range <%s>

Check of Command Verification.High Value for range

Constraint: If given, High_Value_for_Range >= Low_Value_for_range.

CGS–ERR–0102–7 Measurement to be checked <%s>: High Value for Range is defined, Operator has to be ”in_range”

Check of Command Verification.High Value for range

Constraint: If given, Operator must be ”in_range”.

Command Precondition

Check of aggregate Command Precondition in GDUs

CGS–ERR–0106–1 Measurement to be checked <%s> is of type <%s>, Operator has to be either = or /=.

Check of Command Precondition.Operator

Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_... , EGSE_BYTE_STREAM_... or EGSE_STRING_..., Operator must be one of = or /=.

CGS–ERR–0106–2 Measurement to be checked <%s>: Value <%s> has to be convertible to a float value

Check of Command Precondition.Value and Command Precondition.High Value for range

Constraint: If Measurement_to_be_checked is of type EGSE_FLOAT_... , strings given in Value and High_Value_for_Range must be convertible to a float value.

CGS–ERR–0106–3 Measurement to be checked <%s>: Value <%s> has to be convertible to an integer value

Check of Command Precondition.Value and Command Precondition.High Value for range

Constraint: If Measurement_to_be_checked is of type EGSE_INTEGER_..., strings given in Value and High_Value_for_Range must be convertible to an integer value.

CGS–ERR–0106–4 Measurement to be checked <%s>: Value <%s> must be limited to 8 characters (state code)

Check of Command Precondition.Value

Constraint: If Measurement_to_be_checked is of type EGSE_DISCRETE_... , string given in Value must be limited to 8 upper–case characters (state code)

Copyright per DIN 34

CGS–ERR–0106–5 Measurement to be checked <%s>: Operator is ”in_range”, High Value for Range must be defined

Check of Command Precondition.High Value for range

Constraint: If Operator is ”in_range”, both Value and High_Value_for_Range must ge given.

CGS–ERR–0106–6 Measurement to be checked <%s>: High Value for Range <%s> must be equal or greater then Low Value for Range <%s>

Check of Command Precondition.High Value for range

Constraint: If given, High_Value_for_Range >= Low_Value_for_range.

CGS–ERR–0106–7 Measurement to be checked <%s>: High Value for Range is defined, Operator has to be ”in_range”

Check of Command Precondition.High Value for range

Constraint: If given, Operator must be ”in_range”.

PUS Structure ID

CGS–ERR–0109 Mandatory value is not defined.
(Value is mandatory because it exists no reference from a MEASUREMENT_LIST_TABLE end item to the actual end item, where the value for ”Measurement List Table.Fexible Id” is not empty.)

Check of T_STRUCT_ID.F_STRUCTURE_ID

Constraint: In case T_STRUCT_ID is empty, a reference **“**T_MEAS_LIST_TABLE. Reference_to_Structure_Id“ to the actual enditem must exist within scope for an enditem of type MEASUREMENT_LIST_TABLE, where the value for T_MEAS_LIST_TABLE. Flexible_ID is not empty.

Acquisition Data Unit (ADU)

Check of end item types:
PUS_ADU_DESCRIPTION
CCSDS_ADU_DESCRIPTION
UNSTRUCTURED_ADU_DESCRIPTION
STRUCTURED_ADU_DESCRIPTION

LOCATION :	T_DATA_BUF_LAYOUT_END_ITEMS. **Location**
LOC:	{T_REF_TO_STRUCT_ID. **Offset_of_Meas_in_Bits** + (Bits needed for previous measurements contained in referenced Measm List)} aligned to bit/byte/word boundary acc. to T_REF_TO_STRUCT_ID.**value_alignment**
VALUE_SIZE:	T_RAW_VALUE_SIZE_IN_BITS.**Raw Value Size in Bits** or, for Byte Stream Measurements: T_RAW_VALUE_SIZE_IN_BYTES.**Raw Value Size in Bytes** * 8
Packet Length Field:	T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field**
N:	Number of entries (measurement values) in the data buffer
Buffer Length:	T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global_length**

Copyright per DIN 34

CGS–ERR–0111–1 ”Offset to Header in Bits” <%s> + ”Length of ID in Bits” <%s> is greater than ”Global Length.Global Length” <%s>

Check of UNSTRUCTURED_ADU_DESCRIPTION

Constraint: The Offset for the Structure ID (i.e. T_REF_TO_STRUCT_ID. Offset_to_Header_in_Bits) + the number of bits for the Structure ID (i.e. T_REF_TO_STRUCT_ID. Length_of_ID_in_Bits) must not exceed the Length of the Data Buffer:

CGS–ERR–0111–2 ”Offset to Header in Bits” <%s> + ”Length of ID in Bits” <%s> is greater than ”CCSDS Primary Header.Packet Length” <%s>

Check of CCSDS_ADU_DESCRIPTION and PUS_ADU_DESCRIPTION

Constraint: The Offset for the Structure ID + the number of bits for the Structure ID must not exceed the Packet Length.

CGS–ERR–0112–1 For measurements referenced from PUS_STRUCTURE_ID <%s>: Location(last measurement) <%s> + ”Raw Value Size”(last measurement) <%s> – 1 is greater than ”Global Length.Global Length” <%s> * 8

Check of UNSTRUCTURED_ADU_DESCRIPTION

Constraint: Any measurement value to be contained in the ADU (as defined in the PUS_STRUCTURE_ID aggregate) shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOC(last_measurement) + VALUE_SIZE(last_measurement) – 1 <= (Buffer Length) * 8

CGS–ERR–0112–2 For measurements referenced from PUS_STRUCTURE_ID <%s>: Location(last measurement) <%s> + ”Raw Value Size”(last measurement) <%s> – 1 is greater than (”CCSDS Primary Header.Packet Length” <%s> + 1) * 8

Check of CCSDS_ADU_DESCRIPTION and PUS_ADU_DESCRIPTION

Constraint: Any measurement value to be contained in the ADU (as defined in the PUS_STRUCTURE_ID aggregate) shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOC(last_measurement) + VALUE_SIZE(last_measurement) – 1 <= (Packet Length Field + 1) * 8

CGS–ERR–0113 Location <%s> + <%s>.Raw_Value_Size <%s> – 1 is not less or equal than Global Length * 8 <%s>

Check of UNSTRUCTURED_ADU_DESCRIPTION.Measurement End Items.Location

Constraint: Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

Copyright per DIN 34

CGS–ADV–0114 Location <%s> must be greater than previous Location <%s> + <%s>.Raw_Value_Size <%s> – 1

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location, PUS_ADU_DESCRIPTION.Measurement End Items.Location and UNSTRUCTURED_ADU_DESCRIPTION.Measurement End Items.Location

Constraint: Measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$LOCATION_{(i+1)} > LOCATION_{(i)} + VALUE\_SIZE_{(i)} - 1$$

CGS–ERR–0115 If the CCSDS Packet has no secondary header, then the Data Source must not be set to SECONDARY_HEADER.

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Data Source and PUS_ADU_DESCRIPTION.Measurement End Items.Data Source

Constraint: If the CCSDS Packet has no secondary header, then the Data Source must not be set to SECONDARY_HEADER.

Copyright per DIN 34

CGS–ERR–0116–1 Source is ”DATA” and no Datafield Header is defined: Location <%s> + <%s>.Raw_Value_Size <%s> – 1 is not less or equal than (CCSDS_Primary_Header.Packet_Length + 1) * 8 <%s>

CGS–ERR–0116–2 Source is ”DATA” and a Datafield Header is defined: Location <%s> + <%s>.Raw_Value_Size <%s> – 1 is not less or equal than (CCSDS_Primary_Header.Packet_Length – 9) * 8 <%s>

CGS–ERR–0116–3 Source is ”HEADER”: Location <%s> + <%s>.Raw_Value_Size <%s> – 1 is not less or equal than <%s>

CGS–ERR–0116–4 Source is ”SECONDARY_HEADER”: Location <%s> + <%s>.Raw_Value_Size <%s> – 1 is not less or equal than <%s>

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Location

Constraint: Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8
in case the DATA_SOURCE is set to DATA
and no Secondary Header is defined

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field –9) * 8
in case the DATA_SOURCE is set to DATA
and a Secondary Header is defined

LOCATION + VALUE_SIZE – 1 <= 6 * 8
in case the DATA_SOURCE is set to HEADER

LOCATION + VALUE_SIZE – 1 <= 10 * 8
in case the DATA_SOURCE is set to SECONDARY_HEADER

Check of PUS_ADU_DESCRIPTION.Measurement End Items.Location

Constraint: Any measurement value to be contained in the ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field + 1) * 8
in case the DATA_SOURCE is set to DATA
and no Datafield Header is defined

LOCATION + VALUE_SIZE – 1 <= (Packet Length Field –9) * 8
in case the DATA_SOURCE is set to DATA
and a Datafield Header is defined

LOCATION + VALUE_SIZE – 1 <= 6 * 8
in case the DATA_SOURCE is set to HEADER

LOCATION + VALUE_SIZE – 1 <= 10 * 8
in case the DATA_SOURCE is set to
SECONDARY_HEADER

Copyright per DIN 34

CGS–ERR–0117 Source is <%s>: end item type has to be "UNSIGNED_INTEGER" or "SIGNED_INTEGER" for end item <%s> at Location <%s>

Check of CCSDS_ADU_DESCRIPTION.Measurement End Items.Data Source

Constraint: If for any measurement to be contained in the ADU the DATA_SOURCE is set to HEADER or SECONDARY_HEADER, the RAW_VALUE_TYPE of the measurement must be one of the following alternatives:
UNSIGNED_INTEGER
SIGNED_INTEGER

CGS–WRN–0118 Warning: Implementation limit of the number of referenced measurements in a Structured ADU Description (68 in TES) is exceeded. The execution of this item may fail during runtime

Check of STRUCTURED_ADU_DESCRIPTION.Measurement End Items

Constraint: The minimal implementation limit (in TES) of the number of referenced measurements in a Structured ADU Description is 68 in case the flag ADU General Info.All Measurements with Physical Address is set to true.
It exactly depends on the contents of the measurements, in particular the physical address.
If this limit is exceeded then the execution of this item may fail during runtime.

SIMULATED_ADU_DESCRIPTION

Check of end item type SIMULATED_ADU_DESCRIPTION

CGS–ERR–0119–1 Aggregate is not defined

Check of Simulated Data List, Simulated Raw Value List and Simulated Data Global Length

Constraint: The Simulated Data List aggregate must be used if the Referenced ADU designates a STRUCTURED ADU Description.

Constraint: The Simulated Raw Value List aggregate must be used if the type of the Referenced ADU is UNSTRUCTURED or CCSDS_PACKET.

Constraint: The Simulated Data Global Length aggregate must be used if the Referenced ADU is UNSTRUCTURED or CCSDS_PACKET.

CGS–ERR–0119–2 Length <%s> is not less or equal than Packet Length Field + 1 <%s> of the referenced ADU Description

Check of Simulated Data Global Length.Global Length
Reference is a CCSDS_ADU_DESCRIPTON

Constraint: Actual length of Simulated ADU must be less than or equal to the length of the referenced ADU,
i.e. if referenced ADU is a CCSDS ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_CCSDS_HEADER_DESCRIPTION.**Packet Length Field** + 1

Copyright per DIN 34

CGS–ERR–0119–3 Length <%s> is not less or equal than Global Length <%s> of the referenced ADU Description

Check of Simulated Data Global Length.Global Length
Reference is a UNSTRUCTURED_ADU_DESCRIPTON

Constraint: Actual length of Simulated ADU must be less than or equal to the length of the referenced ADU,
or if referenced ADU is an unstructured ADU then:
Sim–ADU → T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length** <=
Ref–ADU → T_DATA_BUFF_LAYOUT_GLOB_LENGTH.**Global Length**

CGS–ERR–0124 Location <%s> + Length <%s> – 1 is not less or equal than Global Length * 8 <%s>

Check of Simulated Raw Value List.Location

Constraint: Any simulated measurement value in an Unstructured or CCSDS Simulated ADU shall be entirely located within the bounds of the data buffer, i.e., the following condition shall be satisfied:

LOCATION + VALUE_SIZE – 1 <= (Buffer Length) * 8

with

LOCATION : T_SIMULATED_RAW_VALUE_LIST. **Location**
VALUE_SIZE: T_SIMULATED_RAW_VALUE_LIST.**Length**
Buffer Length: T_SIMULATED_DATA_GLOBAL_LENGTH.**Global Length**

CGS–ERR–0125 Location <%s> must be greater than previous Location <%s> + Length <%s> – 1

Check of Simulated Raw Value List.Location

Constraint: Further, measurement Values shall not overlap one another, that is:
For any measurement i (i = 1 .. N–1)

$$LOCATION_{(i+1)} > LOCATION_{(i)} + VALUE\_SIZE_{(i)} - 1$$

with

LOCATION : T_SIMULATED_RAW_VALUE_LIST. **Location**
VALUE_SIZE: T_SIMULATED_RAW_VALUE_LIST.**Length**
N: Number of entries (measurement values) in the data buffer

CGS–ERR–0126 Row <%s>: Attribute is NULL

Check of Simulated Raw Value List.Hexa Raw Value

Constraint: If Raw Value in Hex is TRUE then Hexa Raw Value must be defined.

Copyright per DIN 34

CGS–ERR–0127 Row <%s>: Attribute is NULL

Check of:
Simulated Raw Value List.Integer Raw Value
Simulated Raw Value List.Unsigned Integer Raw Value
Simulated Raw Value List.Float Raw Value
Simulated Raw Value List.Bytestream Raw Value

Constraint: The types of values specified in a Simulated ADU must be compatible with the types of the measurements at corresponding locations in the referenced ADU, i.e., the specified simulated data values (in aggregates T_SIMULATED_DATA_LIST or T_SIMULATED_RAW_VALUE_LIST) shall be:

– either **Integer Raw Value** or **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_INTEGER_MEASUREMENT
– **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_DISCRETE_MEASUREMENT
– **Float Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_FLOAT_MEASUREMENT
– **Byte Stream Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_BYTESTREAM_MEASUREMENT

CGS–ERR–0128 Row <%s>: Is not compliant to the length

Check of Simulated Raw Value List.Hexa Raw Value

Constraint: (1) The size of the hexadecimal string depends on the **Length** attribute above.
It must be:
(**Length** / 4) characters if (**Length** mod 4) = 0, or
(**Length** / 4) + 1 characters if (**Length** mod 4) <> 0
(2) The corresponding binary value must fit into **Length** bits, (i.e. only ’0’ and ’1’ are allowed for 1 bit, ’0’ to ’3’ for 2 bits, ’0’ to ’1F’ for 5 bits, etc).

CGS–ERR–0129 Row** [number]**: Is not compliant to the length

Check of Simulated Raw Value List.Integer Raw Value, Simulated Raw Value List.Unsigned Integer Raw Value and Simulated Raw Value List.Float Raw Value

Constraint: The specified value must be within the range defined by the Length attribute, e.g. if Length =4 bits, then value range is: –8 .. 7.

Constraint: The specified value must be within the range defined by the Length attribute, e.g. if Length =3 bits, then value range is: 0 .. 7.

Constraint: The corresponding Length must be 32; specified value must be 32 bits long.

Copyright per DIN 34

CGS–ERR–0132 The number of entries exceeds the number of measurement end items in the Measurement End Item List

Check of Simulated Data List

Constraint: For any given Structured Simulated ADU (i.e. when type of referenced ADU = 'STRUCTURED'), the number of entries in the Simulated Data List (aggregate T_SIMULATED_DATA_LIST) must not exceed the number of measurement end items in the Measurement End Item List (aggregate T_MEASUREMENT_END_ITEM_LIST) of the referenced ADU.

CGS–ERR–0133 Row <%s>: Attribute is NULL

Check of:
Simulated Data List.Integer Raw Value
Simulated Data List.Unsigned Integer Raw Value
Simulated Data List.Float Raw Value
Simulated Data List.Bytestream Raw Value

Constraint: The types of values specified in a Simulated ADU must be compatible with the types of the measurements at corresponding locations in the referenced ADU, i.e., the specified simulated data values (in aggregates T_SIMULATED_DATA_LIST or T_SIMULATED_RAW_VALUE_LIST) shall be:

- either **Integer Raw Value** or **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_INTEGER_MEASUREMENT
- **Unsigned Integer Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_DISCRETE_MEASUREMENT
- **Float Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_FLOAT_MEASUREMENT
- **Byte Stream Raw Value** if the corresponding measurement in the referenced ADU is of type EGSE_BYTESTREAM_MEASUREMENT

GDU_DESCRIPTION_LIST

Check of end item type GDU_DESCRIPTION_LIST

CGS–ERR–0134 Row <%s>: End item <%s> must actually have a parameter list with all parameters being optional, i.e. having default values

Check of Stimuli List.End Item Reference

Constraint: If defined, the parameters of the referenced end item must all be optional with assigned default values.

Parameter related Checks

This checks are defined for end item types having Formal Parameters.

CGS–ERR–0135 <%s> is not conform to a Parameter Name in aggregate Formal Parameters

Check of
List of Parameters

Copyright per DIN 34

Parameter Decalibration Curve Type
Parameter Analog Decalibration Point Pairs
Parameter Analog Calibration Coefficients
Parameter Analog Point Pairs
Parameter Discrete Decalibration
Parameter Raw Value Type
Parameter Raw Value Size in Bits
Parameter Raw Value Size in Bytes
Parameter Integer Raw Value Range
Parameter Unsigned Integer Raw Value Range
Parameter Float Raw Value Range
Parameter Double Float Raw Value Range
Parameter Integer Engineering Range
Parameter Float Engineering Range
Parameter Double Float Engineering Range
Parameter Engineering Unit
String Parameter Definition

Constraint: This is the name of the parameter (same as in aggregate T_FORMAL_PARAMETERS).

CGS–ERR–0136 Parameter <%s>: number of point pairs out of range 2..20

Check of Parameter Analog Decalibration Point Pairs and Parameter Analog (Calibration) Point Pairs

Constraint: Multi–record aggregate. Minimum: 2 , maximum: 20 point pairs per parameter, max. 255 parameter

CGS–ERR–0137 Parameter <%s>: Value <%s> out of Raw Value Range <%s>...<%s>

Check of Parameter Analog Decalibration Point Pairs.Raw Value and Parameter Analog (Calibration) Point Pairs.Raw Value

Constraint: Value must be within the raw value range specified for the particular parameter.

CGS–ERR–0138–1 Parameter <%s>: Value <%s> out of Integer Engineering Range <%s>...<%s>

CGS–ERR–0138–2 Parameter <%s>: Value <%s> out of Float Engineering Range <%s>...<%s>

CGS–ERR–0138–3 Parameter <%s>: Value <%s> out of Double Float Engineering Range <%s>...<%s>

Check of Parameter Analog Decalibration Point Pairs.Engineering Value and
Parameter Analog (Calibration) Point Pairs.Engineering Value

Constraint: Value must be within the engineering value range specified for the particular parameter.

CGS–ERR–0139 Parameter <%s> shall be of type string

Check of String Parameter Definition

Constraint: The specified parameter shall be of type ’string’, i.e. its SW Type must be = ’STRING_TYPE’ or ’BYTE_STRING_TYPE’

Copyright per DIN 34

CGS–ERR–0140 Parameter <%s>: Point pairs describe not a monotonous function

Check of Parameter Analog Decalibration Point Pairs and Parameter Analog (Calibration) Point Pairs

Constraint: For any given parameter, the sequence of N Eng.Values must be monotonic over the specified eng. value range i.e.

$ENG\_VALUE_i < ENG\_VALUE_{i+1}$

or $ENG\_VALUE_i > ENG\_VALUE_{i+1}$ for i=1 .. N–1

For any given parameter, the sequence of N Raw Values must be monotonic over the specified raw value range i.e.

$RAW\_VALUE_i < RAW\_VALUE_{i+1}$

or $RAW\_VALUE_i > RAW\_VALUE_{i+1}$ for i=1 .. N–1

CGS–ERR–0146–1 Parameter <%s>: Raw Value Type <%s> does not correspond to SW type <%s> in Formal Parameter

Check of Parameter Raw Value Type.Raw Value Type

Constraint: The allowed Raw Value Type depends on the **SW type** of the particular parameter as indicated in the following table:

SW Type of Formal Parameter	**Allowed Raw Value Type**
INTEGER_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
UNSIGNED_INTEGER_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
REAL_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
	FLOAT
LONG_REAL_TYPE	SIGNED_INTEGER
	UNSIGNED_INTEGER
	DOUBLE_FLOAT
STATE_CODE_TYPE	UNSIGNED_INTEGER
STRING_TYPE	BYTE_STREAM

CGS–ERR–0146–2 Parameter <%s>: Value <%s> out of range

Check of Parameter Raw Value Size in Bits.Raw Value Size in Bits

Constraint: The allowed Raw Value Size depends on the **Raw Value Type** as indicated in the following table:

Raw Value Type	**Raw Value Size in Bits**
SIGNED_INTEGER	1 .. 32
UNSIGNED_INTEGER	1 .. 32
FLOAT	32
DOUBLE_FLOAT	64
BYTE_STREAM	N/A

Copyright per DIN 34

CGS–ERR–0148 Parameter <%s>: High Value <%s> is not greater than Low Value <%s>

Check of:
Parameter Integer Raw Value Range.High Value
Parameter Unsigned Integer Raw Value Range.High Value
Parameter Float Raw Value Range.High Value
Parameter Double Float Raw Value Range.High Value
Parameter Integer Engineering Range.High Value
Parameter Float Engineering Range.High Value
Parameter Double Float Engineering Range.High Value

Constraint: Low Value < High Value

CGS–ERR–0150 Parameter <%s>: Value must be equal or greater than 0 for an unsigned integer parameter

Check of Parameter Integer Engineering Range.Low Value

Constraint: If the SW Type of Formal Parameter is UNSIGNED_INTEGER_TYPE then the Low Value must be equal or greater than 0.

CGS–ERR–0151 Statecode <%s> is not unique in parameter <%s>

Check of Parameter Discrete Decalibration.State Code

Constraint: must be unique for one given stimulus

CGS–ERR–0152 Parameter <%s>: Number of calibration values is not less or equal than 2 ** raw value size in bytes

Check of Parameter Discrete Decalibration

Constraint: must be <= 2**Parameter Raw Value Size in Bits

CGS–ERR–0153 Parameter <%s>: Too many calibration raw value/state code pairs (>1024)

Check of Parameter Discrete Decalibration

Constraint: Maximum: 1024 raw value/state code pairs per parameter, max 255 parameter

CGS–ERR–0154 Too many parameters (> 255)

Check of Formal Parameter for end item types:
EGSE_PREDEFINED_TC
PUS_TC
EGSE_BINARY_PACKET
SWOP_COMMAND

Constraint: Up to 255 formal parameters are allowed.

APID

Check of end item type APID

Copyright per DIN 34

CGS–ERR–0155 Combination of Source CCSDS End Point <%s> and Destination CCSDS End Point <%s> is not unique in configuration scope for end items: <%s>

Check of Apid Table.Source CCSDS End Point and Apid Table.Destination CCSDS End Point

Constraint: The combination of the two attributes Source CCSDS End Point and Destination CCSDS End Point must be unique within the configuration scope.

CGS–ERR–0156 Combination of Application ID <%s> and Type <%s> is not unique in configuration scope for end items: <%s>

Check of Apid Table.Application ID and Apid Table.Type

Constraint: The combination of the two attributes Application ID and Type must be unique within the configuration scope.

SWOP_COMMAND

Check of end item type SWOP_COMMAND

CGS–ERR–0158–1 Value must be TRUE

Check of CCSDS Second Header.Checksum Indicator

Constraint: Checksum Indicator in aggregate CCSDS Second Header must be TRUE.

CGS–ERR–0158–2 Value must be CCSDS_SYSTEM_COMMAND_PACKET or CCSDS_ESSENTIAL_COMMAND_PACKET or CCSDS_PAYLOAD_COMMAND_PACKET

Check of CCSDS Second Header.Packet Type

Constraint: Packet Type in CCSDS Second Header must be ”System_Command”, ”Essential_Command”, ”Payload_Command” or “Memory_Load”.

CGS–ERR–0158–3 Value must be NO_TIME_FIELD or TIME_OF_PACKET_GENERATION

Check of CCSDS Second Header.Time ID

Constraint: Time ID in CCSDS Second Header must be NO_TIME_FIELD or TIME_OF_PACKET_GENERATION.

SWRU

Check of end item type SWRU

CGS–ERR–0161 Initialization and Exchange AP shall not refer to the same AP.

Check of General.Initialization AP

Constraint: Initialization and Exchange APs shall not refer to the same AP.

Discrete Calibration (continuation)

see also NO TAG and NO TAG

Copyright per DIN 34

CGS–ERR–0162 Low Value <%s> not less than corresponding high value <%s>

Check of Discrete Calibration.Calibration Raw Value (Low Value) and Discrete Calibration.High Value of Raw Value Range

Constraint: The relation `Low_Raw_Value < High_Raw_Value` must be fulfilled for each record, if `High_Raw_Value` is given.

CGS–ERR–0163 Raw Value Calibration Ranges overlap: <%s>

Check of Discrete Calibration.Calibration Raw Value (Low Value) and Discrete Calibration.High Value of Raw Value Range

Constraint: The defined raw value ranges must not overlap

CGS–ERR–0164 State Code Value ”OTHER” is reserved and may not appear as a Calibration State Code

Check of Discrete Calibration.Discrete Calibration State Code

Constraint: The state code value ’OTHER’ is reserved and may not appear as a valid calibration state code.

CGS–ERR–0165 State Code value <%s> is not conform to UCL syntax

Check of

Parameter Discrete Decalibration.State Code
Discrete Calibration.Discrete Calibration State Code
Undefined Values State Code.Undefined Values State Code
Discrete Conditions.Value
Expected Value.Expected Value
Initial Discrete Value.Initial Discrete Value

Constraint: The state code value must be conform to the UCL syntax.

Copyright per DIN 34

N-2 List of Single Enditem Checks (Check MDB Item)

The Check_MDB_Item program is implemented for the following enditem types:

MEAS	EGSE_xxx_MEASUREMENT EGSE_xxx_SW_VARIABLE EGSE_xxx_DERIVED_VALUE
ADU	xxx_ADU_DESCRIPTION
GDU	EGSE_PREDEFINED_TC EGSE_BINARY_PACKET EGSE_ANALOG_STIMULUS EGSE_DISCRETE_STIMULUS
SYNOPT	WDU_GROUND_SYNOPTIC_DISPLAY

The checks as listed in the following table are defined:

Copyright per DIN 34

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	MEAS	RANGE	ERROR	Danger Limits not in Engineering Range
C2	MEAS	RANGE	ERROR	Nominal Limit not in Engineering Range
C5	MEAS	RANGE	ERROR	Danger High Limit < Danger Low Limit
C10	MEAS	RANGE	ERROR	Nominal High Limit < Nominal Low Limit
C12	MEAS	RANGE	ERROR	Danger High Limit < Nominal High Limit
C13	MEAS	RANGE	ERROR	Danger Low Limit > Nominal Low Limit
C14	MEAS	RANGE	ERROR	Expected Value not defined as Statecode in Calibration
C15	MEAS	RANGE	WARN	Raw value range does not include 0 – dangerous for calibration of uninitialized values
C20	MEAS	RANGE	ERROR	Low Value of Engineering Range >= High Value of Engineering Range
C21	MEAS	RANGE	ERROR	Low Value of Engineering Range > Initial Value > High Value of Engineering Range
C22	MEAS	RANGE	ERROR	Alarm Counter range 1..10
C23	MEAS	RANGE	ERROR	Condition: In_range: Value_2(High_Value) is lower than Value_1 (Low_Value)
C1	MEAS	CALIB	ERROR	Wrong calibration definitions
C2	MEAS	CALIB	ERROR	Raw Value in discrete calibration > 2** raw_value_size_in_bits
C7	MEAS	CALIB	ERROR	OTHER must not be used as a normal calibration code
C8*	MEAS	CALIB	ERROR	Condition: Referenced Statecode not in calibration state code list
C1	MEAS	MANDAT	WARN	Initial Value is 0 (in DB: undefined or 0) – Might be unwanted
C2	MEAS	MANDAT	ERROR	Engineering Range: Optional. Default is min–value .. max_value Raw Value Range: Optional. Default is min–value .. max_value Calibration Raw Value Type / Raw Value Size
C3*	MEAS	MANDAT	ERROR	If calibration is given, raw value definition is mandatory for loading sw – even if not used in GWDU (applies to discrete sw variables/derived values only)
C3	MEAS	REFER	ERROR	Undefined/Wrong References in Derived Value Expressions
C4	MEAS	REFER	ERROR	Conditions: Wrong/Undefined References in Conditions Referenced AP must not have parameters

Copyright per DIN 34

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	MEAS	REFER	ERROR	Reference for Action/Message invalid –Reference in Monitoring Action Descriptions: AP,GDU,GDU_List – Reference in Monitoring Action Descriptions: AP without Parameter – Reference in Monitoring Message Descriptions: EGSE_User_Message
C5	MEAS	REFER	ERROR	Conditions: Referenced Enditem Type not compatible with Comparator
C1	MEAS	MISC	ERROR	Danger Limits cannot be defined without nominal limits
C2*	MEAS	MISC	ERROR	Engineering Unit is not valid
C1	GDU	RANGE	ERROR	Number of Formal Parameters must be <= 255
C2	GDU	RANGE	ERROR	For each Formal Parameter there must be an entry in List_of_Parameters
C3	GDU	RANGE	ERROR	For each Parameter the raw value size must conform to Number of Bits in List_of_Parameters
C4	GDU	RANGE	ERROR	Parameter Values: Location + Size –1 <= Packet length*8
C5	GDU	RANGE	ERROR	Predefined Values: location + number_of_bits – 1 <= (number of bytes in packet)*8
C6	GDU	RANGE	ERROR	Length in CCSDS header: 9 <= length <= 4095 resp. 1 <= length <= 4095 (if no 2nd header)
C7	GDU	RANGE	ERROR	Maximum Length acc. to Parametertype”)); 1 <= length <= 31 for statecodes 8 <= length <= 4096 for strings 1 <= length <= 32 for integer length = 32 for pathnames length = 32 for floats (real) length = 40 for time
C8	GDU	RANGE	ERROR	GDU Verification: Time value is negative –> Verification will be ignored
C9	GDU	RANGE	ERROR	GDU Verification: In_Range, but high value < low value —> Verification will be ignored
C1	GDU	MANDAT	ERROR	SAS Reference in General Info
C2	GDU	MANDAT	ERROR	CCSDS Primary Header Fields
C3	GDU	MANDAT	WARN	CCSDS Secondary Header: Packet ID not defined: is set to GDU SID: Might be unwanted
C4	GDU	MANDAT	ERROR	Analog/Discrete Stimulus must have exactly 1 Parameter: Is parameter definition compiled?
C5	GDU	MANDAT	ERROR	Bitstream Layout Attributes are all mandatory

Copyright per DIN 34

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C6	GDU	MANDAT	ERROR	Definition Attributes are all mandator for IN-TEGER_DEFINITION, FLOAT_DEFINITION, BINARY_DEF.
C1	GDU	CALIB	ERROR	Analog Decalibration: Curve Type is POINT_PAIRS, but no point pairs given
C2	GDU	CALIB	ERROR	Decalibration defined, but no raw value / Raw value type does not conform to decalibration
C3	GDU	CALIB	ERROR	Engineering Range for Parameter not valid / not defined
C4	GDU	CALIB	ERROR	Raw Value Range for Parameter not valid / not defined
C5*	GDU	CALIB	ERROR	Condition: Referenced Statecode not in calibration state code list
C1	GDU	REF	ERROR	GDU Verification: Reference to wrong enditem type
C2	GDU	REF	ERROR	GDU Verification: Referenced Enditem Type not compat-ible with Comparator
C1	GDU	MISC	ERROR	Type of Parameters must be of mode IN
C2	GDU	MISC	ERROR	Type of Parameters must be one of STRING,STATE_CODE,INTEGER,UNSIGNED_IN-TEGER,REAL,PATHNAME,TIME
C3	GDU	MISC	ERROR	Parameter is optional, but has no default value
C1	ADU	RANGE	ERROR	Length in CCSDS header: 9 <= length <= 4095 resp. 1 <= length <= 4095 (if no 2nd header
C2	ADU	RANGE	ERROR	Global Length / Bit Locations in Unstructured ADUs : 1<= location <= global length
C1	ADU	MANDAT	ERROR	SAS Reference in General Info
C2	ADU	MANDAT	ERROR	CCSDS Primary Header Fields
C3	ADU	MANDAT	ERROR	CCSDS Secondary Header: Packet ID not defined: is set to ADU SID: Might be unwanted
C1	ADU	REF	ERROR	Referenced measurement not found in selected scope
C2	ADU	REF	ERROR	Referenced item is not of type EGSE_xxx_MEA-SUREMENT
C3	ADU	REF	WARN	ADU Descriptor does not contain any reference to a mea-surement
C1	ADU	MISC	WARN	Overlapping of Bit Positions for measurements in ADU

Copyright per DIN 34

ID	ENDI-TEM CLASS	CHECK CLASS	MSG TYPE	MESSAGE / Description
C1	SYNOPT	REF	ERROR	Referenced variable (measurement) not found in selected scope
C2	SYNOPT	REF	ERROR	Referenced synoptic (for Display Action) not found in selected scope

* => check is currently not implemented

Copyright per DIN 34

O CIS INTERFACE DESCRIPTION

The CIS interface is descibed in the CORBA IDL specification language.

Details can be found in the CGS ICD (see reference in chapter 2, 2.1.3).

Copyright per DIN 34